U.S. Department
of Commerce

National Bureau
of Standards

Computer Science and Technology

NBS Special Publication 500-69

An Analytic Study of a Shared Device Among Independent Computing Systems

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards[1] was established by an act of Congress on March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, and the Institute for Computer Sciences and Technology.

THE NATIONAL MEASUREMENT LABORATORY provides the national system of physical and chemical and materials measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; conducts materials research leading to improved methods of measurement, standards, and data on the properties of materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

Absolute Physical Quantities[2] — Radiation Research — Thermodynamics and Molecular Science — Analytical Chemistry — Materials Science.

THE NATIONAL ENGINEERING LABORATORY provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

Applied Mathematics — Electronics and Electrical Engineering[2] — Mechanical Engineering and Process Technology[2] — Building Technology — Fire Research — Consumer Product Technology — Field Methods.

THE INSTITUTE FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following centers:

Programming Science and Technology — Computer Systems Engineering.

[1]Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Washington, DC 20234.
[2]Some divisions within the center are located at Boulder, CO 80303.

Computer Science and Technology

NBS Special Publication 500-69

An Analytic Study of a Shared Device Among Independent Computing Systems

Alan Mink

Center for Computer Systems Engineering
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, DC 20234

U.S. DEPARTMENT OF COMMERCE
Philip M. Klutznick, Secretary

Luther H. Hodges, Jr., Deputy Secretary

Jordan J. Baruch, Assistant Secretary for Productivity, Technology and Innovation

National Bureau of Standards
Ernest Ambler, Director

Issued November 1980

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer unitilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication.

National Bureau of Standards Special Publication 500-69
Nat. Bur. Stand. (U.S.), Spec. Publ. 500-69, 176 pages (Nov. 1980)
CODEN: XNBSAV

Library of Congress Catalog Card Number: 80-600170

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1980

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402
Price $5.50
(Add 25 percent for other than U.S. mailing)

ABSTRACT

Global queueing network performance models are developed for the increasingly important class of computer networks comprising a number of independent computing systems sharing a single resource. An extensive bibliography and survey of prior work relating to this topic are included. Analytic expressions of performance measures for this class of systems are derived from the general theory of multi-class queueing networks, and new computational algorithms for evaluating them are presented that are memory-space efficient (linear vs. exponential) compared with known algorithms for the general theory. This exact analytic model, called the Shared Central Server Model, incurs approximately the same exponential time complexity in its evaluation as do all models based on the general theory; because of this, a simple heuristic approximate model of this class of systems is also presented that is computationally efficient in both time and space. Modular expansion of this class of systems is investigated using the approximate model, and a useful relationship is derived between the number of additional independent computing systems and the incremental increase in capability of the shared resource required to maintain the existing level of system performance.

Key words: Approximate queueing models; computer architecture; modular expansion analysis; performance evaluation; performance modeling; queueing models; queueing networks.

ACKNOWLEDGEMENT

The author would like to express his sincere thanks to the following people. To Dr. Charles B. Silio, my deepest appreciation for providing the light on the long dark road, and the push over the mountains. To Dr. James H. Pugsley, my sincere thanks for his patience and guidance. To Mr. Joseph R. Singer, my deepest gratitude for his continual understanding, consideration, and support. To Ms. Linda Jenista and Ms. Elizabeth Becker, my thanks for their assistance in preparation of this manuscript. To Dr. Marshall Abrams, Dr. Ira Cotton, Ms. Shirley Watkins, and Dr. Paul Amer, my thanks for their support and understanding during the final phases of this endeavor. To Mr. Tom Giammo, my thanks for his insight and consultation. Last, but not least, to Dr. Mary Catherine McKenna, my deepest appreciation for providing impulses of energy to re-establish momentum when necessary.

---

* Note: This report was submitted as a dissertation in partial completion of the requirements for the degree of Doctor of Philosophy in Electrical Engineering at The University of Maryland.

TABLE OF CONTENTS

APPENDICES

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. Resource Sharing

Resource sharing is an old concept. It exists in every industry and facet of life. The basic motivation for resource sharing is the existence of a scarce resource, caused primarily by economic or physical considerations.

In the computer industry there has been significant interest in distributed processing [IDC 76] and a proliferation of computer networks [COTTON 79, LEUNG 78, MONAHA 79, SPRING 78, WILKES 79]. The primary goal of these schemes is to provide various mechanisms for resources sharing [KAHN 72, ROBERT 70]. In these environments the scarce resources include information and capabilities (programs or processes) as well as devices.

The cost of processing power and primary memory is currently decreasing at a rate of 50% per year, while the costs of communication facilities, secondary memory, peripherals and special purpose devices are decreasing at a rate of 10% per year or less [BBN 79]. LSI techniques and mass production have been the primary causes of these cost reductions. This has resulted in two basic system development strategies that can be generally associated with opposite ends of the computer cost spectrum. These two strategies can be categorized as more-for-the-same-cost and the-same-for-less-cost.

The trend on the upper end of the cost spectrum, is for the cost not to decrease, but to compensate by providing increased speed, capacity, and capability. New generations of more powerful systems are being offered which include parallel architectures based on array and pipeline concepts. Previously their cost would have been prohibitive. As a result, their predecessors of the previous generation are subsequently being offered on the market at discount prices.

On the lower end of the cost spectrum actual cost reductions are being offered. These basically comprise microcomputer and minicomputer systems, as well as some individual

components. With each cost reduction the acquisition of these systems becomes more feasible for an increasing segment of the business community and the general public.

The cost reduction in microcomputer and minicomputer systems as compared to maxicomputer systems has encouraged acquisition of many independent small systems vs. a single large one. Functional and administrative separation, along with the lower price, has also fueled this trend [IDC 76]. But as applications become more sophisticated and complex they tend to require more data from other parts of the organization and/or more computational power. In this environment the small systems can handle the local and overhead processing, allowing the large system to be utilized efficiently to provide computational power and large storage repositories.

Secondary storage systems, peripherals, and special purpose devices are now the scarce resources within and between computing systems, rather than the processors and primary memories as was previously the case. The increasing economies of scale in secondary storage technology has made the concept of a large pooled storage subsystem attractive [WATSON 80]. One of the objectives of a Back-End Network [CHAMPI 80, LAM 79, WATSON 80] is to provide high speed access to global peripherals and storage subsystems [CHAMPI 80, WATSON 80]. The Octopus network at Lawerence Livermore Laboratory and NASA's Skylab network are two examples of this type of architecture [THORNT 80]. Several mass storage subsystems are currently on the market (e.g. IBM 3850 and CDC 38500). A recent mass storage subsystem design from Nippon electronics Co. [SEKINO 79] listed shared use by multiple independent computer systems as one of its primary design criteria. Although large expensive systems have always been a scarce resource, they are now being shared by low cost small systems. Some organizations that had more than one independent computing system (ICS) have integrated them [IDC 76]. Their motivation has been to reduce costs by sharing resources (data, programs, and devices). Other organizations, in similar circumstances, are investigating the feasibility and benefits of embarking on similar integration efforts [IDC 76].

Two popular approaches to performing this type of integration are through a common shared secondary memory system [CDC 75] or a local area network (LAN) [CARPEN 79]. Neither of these approaches excludes the other. The shared secondary memory approach requires a multiplexing device between the ICSs and the shared secondary memory. Some intelligence is

also required either in each ICS or in the multiplexor to handle the synchronization and lock-out mechanisms necessary to accommodate simultaneous access to common objects. If the ICSs are not local, then communication facilities are also required. If the ICSs are within a few kilometers, then this communication facility could be a LAN.

The LAN approach requires a communication medium and a number of interface units, at least one for each device placed on the network. It is desirable for these interface units to have some intelligence, so that the operation of the LAN is kept relatively transparent to each device. A device on the network may be an ICS, a shared secondary memory, or any other device that may be shared or desires to share the devices on the network.

B. Objectives

We have briefly discussed the rationale for resource sharing within and between computer systems. Two approaches that have been used to accomplish resource sharing were also discussed. The resulting architectures of these approaches facilitate modular expansion by allowing the addition of ICSs, as well as making the sharing of data easier. It is our objective in this dissertation to investigate these types of systems which share resources. Although our previous discussion has been primarily concerned with secondary memory as the shared processing resource (SPR), it is applicable to any SPR. Our focus will be on an architecture consisting of a single SPR among a number of ICSs. Our investigation will be concerned with the relationship between the performance of each ICS based on the processing rate of the SPR and the number of ICSs. To accomplish this we will construct a queueing model, and develop the expressions for the desired performance measures.

Little, if any, analysis on this class of architecture has appeared in the literature. Most of the analyses are concerned with the individual subsystems, rather than the overall system. Investigation into the performance evaluation of ICSs [BRANDW 77, BUZEN 71], secondary memory subsystems [CHANG 72, COFFMA 68A, HOOGEN 77], and general communication

subsystems [FRANK 72, KLIENR 64, KOBAYA 77, WONG 78] have and continue to be done. We feel it is important to provide designers and analysts the results of an analysis on this class of architectures, while in addition providing them with useful analytic tools.

By taking this global analysis viewpoint we will not directly take into account the effects caused by various strategies within the subsystems, such as the effects of different communication protocols. We will assume that the communication subsystem (multiplexors, LAN, etc.) has sufficient bandwidth so that it may be disregarded as a bottleneck for performance evaluation purposes [THORTON 80]. Generally these delay times are insignificant when compared to the processing delays of the various devices comprising the system. When these times are significant, then the processing delays of the devices can be extended to incorporate them.

C. Organization of Dissertation

Chapter II discusses some of the related and previous work in the computer queueing preformance evaluation area. Appendix A provides a short review of the mathematics of queueing network theory. Appendices B and C provide a glossary of terms and a definition of mathematical notation used in this dissertation, respectively. Due to the rather extensive use of mathematics in this dissertation the reader is urged to refer to Appendix C whenever unfamilar notation is encountered.

In chapter III a queueing network model is developed for the system architecture we have introduced here, along with relevant performance measures. Efficient computational algorithms are presented for the evaluation of these performance measures.

Previously the evaluation of queueing network models required memory-space and time complexity both growing exponentially with the size of the state-space. The algorithms we develop to evaluate our model require memory-space that grows linearly with the size of the state-

space, although the time complexity still grows exponentially. This provides the designer and analyst the ability to evaluate this model when it has a large state-space if they are willing to invest the computation time. Whereas, previously it may not have been possible due to physical memory-space limitations.

Chapter IV presents an approximate model for this class of architectures that is significantly more efficient in computation time and memory-space complexity than is the exact model of chapter III. The associated performance measures and an efficient computational algorithm to evaluate them are also presented. The results of this approximation are compared to those of the exact model.

The performance measures predicted by this approximate model do result in a varying relative error, which we consider to be within acceptable engineering limits. The efficiency gained in their evaluation is, for most applications, thought to be an acceptable compromise for the error incurred. For situations with extremely large state-spaces, it may be the only analysis method possible. As a result we provide the designer and analyst the capability to use the approximate model to obtain estimates of the performance of a large number of system configurations in a very short period of time. Once a small number of candidate configurations are culled, the exact model may be applied to obtain more accurate performance predictions.

In chapter V both the exact and approximate models are analyzed to obtain relationships between response/delay times as a function of the number of ICSs and the processing rate of the SPR. The analysis of the exact model is shown to produce only an upper bound which is too high to be of any practical use. The analysis of the approximate model yields a very useful and intuitively satisfying relation between the addition of ICSs and the incremental increase in SPR processing rate required to maintain system response time. This result will be useful to designers and analysts when they consider building new systems or augmenting existing systems which are based on this class of architecture. These results are then applied to two design situations to provide examples of the utility of this model.

Chapter VI summarizes the salient results of this dissertation. Some directions for future potential research extensions are also discussed.

II. MODELING CONCEPTS

A. Related Areas

Prior to setting forth the proposed shared processing architecture (SPR) two related areas were investigated. One is the development and status of the theory that is to be applied toward the modeling endeavor. The second is similar situations that have been studied.

Queueing network theory will be applied toward developing our SPR model. Existing queueing network modeling techniques and programming facilities, other than product form, reasonably allow analysis of systems consisting of a few thousand states. These techniques are generally limited by algorithm complexity and machine resources. There is currently no indication that significantly larger state-spaces will be accommodated in the near future [CHANDY 78]. However, systems whose structure conforms to the requirements of a product form solution may accommodate state-space sizes many times larger than other queueing network modeling techniques. In the next section we review the area of queueing network theory.

Modeling requires one to be concerned with two levels of abstraction. The first level is where the analyst specifies a descriptive model by selecting "key" aspects of the actual system. The second level is where the analyst formulates or applies an analytic model to represent the descriptive model. In formulating both these levels of models various simplifying assumptions are generally made. In applying these assumptions the resultant models, on either level, stray from accurately portraying the actual system. In the general literature, as in this dissertation, a first level descriptive model is presented and a second level analytic model is formulated to closely or exactly represent that descriptive model. But the accuracy of the analytic model depends on how well the descriptive model represents the actual system.

In certain situations this presents a dilemma to the analyst, whether to formulate an inaccurate descriptive model for which an analytic model can provide an exact solution; or to formulate an accurate descriptive model for which an approximate analytic model can provide an inaccurate

solution. Both of these approaches yield inaccurate results for the actual system. The more correct approach remains an open question, to be handled on a case-by-case basis.

The approach chosen by a given analyst depends on many factors, such as time, analytic tools and techniques both familar and available, as well as the level of confidence in them. Chandy [CHANDY 78] suggests the analyst selection criteria are, in descending order of importance, (1) solution speed, (2) credibility, and (3) degree of accuracy. Based on this ordering it seems that analysts will sacrifice accuracy for quick results. This should be further qualified. An analyst will sacrifice accuracy in return for a quick solution if it will provide insight into the behavioral trends of the actual system. Therefore, one may conclude that approximate models that provide a fast solution and/or more closely represent a faithful descriptive model of an actual system are always in demand.

The major limitations of current queueing network theory can be placed in two categories, size and structure. Size limitations are concerned with the time and memory-space complexity required to obtain solutions to the models of systems which have large state spaces. Structure limitations are concerned with systems whose operational structure does not conform to the basic assumptions and requirements necessary to be modelled by queueing network theory. As a result of these queueing network theory limitations various approximations to model systems which suffered from one or more of these limitations have been proposed. Depending on the actual system and the limitations one is attempting to overcome, these approximate models produce varying degrees of success and utility. In section C we review some of these approximation techniques.

Although no previous known work has been attempted for our SPR architecture, a somewhat similar situation is the study of memory interference (MI) [BASKET 76, BHANDA 73, BURNET 70, HOOGEN 77, MCCRED 73, OSTERW 72, RAU 79, SMITH 77]. Briefly, this environment consists of n processors sharing m primary memory modules. The main analysis endeavor is to determine the resultant effective memory bandwidth available to the processors. This class of descriptive models differs in three primary aspects from the descriptive operational structure of our SPR architecture.

First, in MI the processor and memory are tightly coupled such that processors operate directly out of primary memory, and their interactions occur within a few clock cycles. In contrast, the SPR is a separate device which is loosely coupled, and interactions require hundreds or thousands of clock cycles. For MI, any delay caused by interference from another processor attempting to access the same primary memory module results in the blocked processor becoming inactive. Work cannot progress without the information from primary memory. The exceptions to this are machines that have have buffered look-ahead and/or prefetch environments. This resulting inactivity applies as well to multiprogrammed processors, since the context cannot be switched due to the nonexistence of a quiescent state and the prohibitive overhead which would be incurred. For the SPR, the processor formulates a request for service to the SPR according to some type of message protocol. Even if the SPR is not busy and the request does not encounter any additional delays other than that required to perform the actual service, the processor expects a relatively large amount of time to elapse before receiving the response. During this time the processor can then attempt to continue processing (i.e. overlap) or if multiprogrammed can switch context and proceed to process another job.

Second, MI is concerned with the interference occuring at the access to the individual memory modules and is not concerned with the job flow or direct delays to any other devices. The SPR architecture is concerned with the job flow and the direct delays a job incurrs as it flows through the system. The integrity of maintaining the proper flow paths is a prime consideration of the SPR model. MI has no flow path, other than the implied processor to primary memory and return, and has no reason to maintain one. Therefore, after the primary memory services the processor's request there is no distinction or accounting as to which processor the job returns to.

Third, MI generally assumes at least two or more primary memory modules and two or more processors, each of which are identical. The SPR architecture assumes only a single SPR and one or more processors, each of which may be distinctly different and have a number of peripheral processors.

The basic approach in constructing MI analytic models has been as follows. Assume a probability distribution for the primary memory access pattern and a relation between the processor and memory cycle time. From these derive the probability of a processor making a

request to each of the memory modules. Then apply combinatorics to weight these probabilities to determine the mean number of busy memory modules and the expected unit execution rate of the processors.

A different approach was used by McCredie [MCCRED 73] which was based on queueing network theory. There are many differences between McCredie's model and the SPR architecture which do not make this model applicable. A single class of jobs is assumed and therefore, no flow path integrity between processors can be maintained. There are a multiple number of identical shared devices and only one job per processor is allowed. McCredie relied on Buzen's algorithm [BUZEN 71] to evaluate this model, which is efficient since only a single job class was involved.

B. Queueing Networks

Queueing network models were originally developed as an aid to the management sciences for "jobshop" flow problems. This specific class of problems consisted of those in which various work requests flow through a network of service centers, forming waiting lines (queues) at each depending upon the density of traffic. As computing systems became more sophisticated, by distributing functions in channels and controllers, it was recognized that the executing characteristics (flow) of a job could be thought of as migrating between these service centers (CPUs, channels, controllers, etc.) and, therefore, could be modelled using queueing networks.

There are two basic research approaches to queueing network theory, probabilistic and algebraic. The probabilistic or decomposition [DISNEY 73] approach decomposes the network into subnetworks, solves the stochastic flow (arrival and departure processes) of each subnetwork independently, and then recombines the results to produce the overall stochastic flow. This approach has the advantage of allowing quite general rules concerning the arrival processes and service distributions, which may be non-markovian in nature. The main disadvantage is a consequence of their general stochastic nature, which usually results in intractable equations yielding no closed form solution. Currently, solutions are available for networks consisting of only two or three service centers.

The algebraic [WALLAC 73] approach represents the state probabilities as a set of homogeneous algebraic equations to be solved. The state is a vector description of the distribution of jobs among the service centers of the network. The main advantage of the algebraic approach is the ability to handle networks with a large number of service centers. The main disadvantage of this approach is the restrictive assumptions on arrival processes and service distributions, generally Markovian.

The algebraic approach has been applied using various techniques. A separation of variables technique has been applied by Jackson [JACKSO 63]. Some numerical evaluation techniques have been attempted. This approach expands the system state-space description so that the model may remain Markovian, but also increases the size of the state-space. The equilibrium state probability equations are formulated and then solved numerically using techniques such as the relaxed Jacobi iteration [WALLAC 66] method and more recently the Gauss Seidel [GAVER 76] method. Analytic solutions represent a special subclass of Markovian networks, but yield very efficient computations due to their structural properties. Numerical solutions on the other hand can handle any Markovian network, but are limited by the size of their state space. In this dissertation we are mainly concerned with analytic solutions to the algebraic approach of modeling systems.

A general queueing network consists of a set of service centers arbitrarily connected, each with a queue and an arbitrary but fixed number of servers. The network is referred to as closed if no new jobs arrive or leave, but a constant number continuously circulate. If arrivals and departures are allowed, permitting the overall number of customers to vary, the network is referred to as open. Operationally, a job arrives at a service center and is placed in a queue, until a server, according to some scheduling policy, is available to provide the required service. Upon completion of service, the job transits, with no delays and a constant known routing probability, to another service center. This sequence is repeated at each service center.

A queueing network is specified by the number of service centers, the number of servers at each center, their service time probability distributions and scheduling policy, a probability transition matrix, the number of jobs in the network (if a closed network) or an arrival time probability distribution (if an open network). The network equilibrium state probability

distribution is computed from the above parameters and is subsequently used to compute various performance measures. Performance measures of general interest include mean queue length, busy probability, and throughput (jobs/unit-time).

The "classical" algebraic solution technique, in general, requires simplifying assumptions to allow the system to be modeled as a Markovian network, having a managable solution. The service and arrival processes of jobs at each service center are assumed to be statistically independent and identically distributed (iid). The service process is assumed to be exponentially distributed. The arrival process is assumed to be Poisson, which implies the time interval between consecutive arrivals has an exponential distribution. The scheduling policy is assumed to be first-come first-served (FCFS). This allows the probabilistic flow rate into and out of a state to be expressed as a simple linear function of time, the mean service rate, and the mean arrival rate. From the resultant expressions for the probabalistic flow rates a set of simultaneous state balance equations is formed, and a product form solution is assumed which eventually reduces to a set of simultaneous linear equations. Appendix A provides a brief review of this solution technique.

Jackson [JACKSO 57] presents one of the earliest works on solving the equilibrium state probabilities of an open queueing network. A fairly general network is assumed. It consists of a set of service centers each containing an arbitrary number of servers. Arrivals into the network follow a Poisson process whose service times are iid exponential distributions, and service is rendered on a FCFS basis. A solution for the equilibrium state probability distribution is presented. Jackson points out the similarity between the form of this solution and that of an elementary single service center with multiple servers, under the same arrival and service assumptions.

Jackson [JACKSO 63] later extended this model by incorporating arrival and service time distributions which are functions of the queue length. In addition he generalized the upper and lower limits on the number of customers in the network, which were previously infinity and zero, respectively. He introduced the concepts of "triggered arrivals" and "service deletions" to accomplish this. Triggered arrivals occur when the total number of customers drop below a specified threshold, thus triggering immediate arrivals of new customers to replace those that left. Service deletions occur when the length of a queue exceeds some maximum threshold. New customers arriving at this queue are given zero service time and continue on their routing.

Gordon and Newell [GORDON 67A] independently produced a result similiar to Jackson's [JACKSO 63] in that they derived the equilibrium state probabilities for a closed queueing network. Gordon and Newell represent this closed job flow system as an irreducible Markov process, consisting of a constant number of customers whose service time distributions are exponential. They present the state balance (difference) equations for this system. By assuming the solution is of a product-form and utilizing a separation of variables technique, they obtain a set of linear simultaneous equations of the form $E = E[P]$. Since [P] is the transition probability matrix, which is stochastic, a solution to the simultaneous equations exists. The solution to these simutaneous equations can then be substituted back into the assumed product-form solution for the equilibrium state probabilities. The product form solution incorporates a normalization constant whose purpose is to force these product terms to be proper probabilities that sum to unity. Thus the normalization constant can be solved for by summing the product-form terms over the entire state space. The state space grows exponentially, $O[(K+1)^s]$, with the number of jobs, K, and the number of service centers, s, in the network. Therefore, this presents a nontrivial computational requirement.

A simple, nontrivial case of a closed queueing network model is that of the Central Server model. Buzen's [BUZEN 71] work on the central server model is among the initial applications of closed queueing network theory to computer systems. The model consists of a CPU (the central server) and a set of peripheral processors (PPUs) which service a set of continually circulating jobs, see Figure III-2 of chapter III. The behavioral characteristics of a job are as follows. A job requests service from the CPU. If the CPU is busy, the job must wait in a FCFS queue. Once the CPU service request is satisfied, the job then transits to one of the PPUs or back to the CPU with a fixed probability. After service is completed at the PPU, the job proceeds with probability one to the CPU. The probability transition matrix, [P], for the central server model consists of a single nonzero row and column, see Figure A-2 of Appendix A. Based on Gordon and Newell's technique the solution to the resulting set of simultaneous equations is obtainable by inspection. Substituting this solution into the product form yields the expression for the equilibrium state probabilities; one must still solve for the normalization constant, which is a nontrivial task.

Buzen introduced an efficient iterative procedure to solve for the normalization constant. Instead of growing exponentially, the computational complexity of Buzen's algorithm grows as O[Ks], where K is the number of jobs and s is the number of service centers in the network. Buzen further derived expressions for the busy probability, throughput and mean queue length of each service center (CPU and PPUs). Buzen also developed a similar central server model which allowed the mean service rates to be arbitrarily dependent on queue lengths, but still required exponential distributions.

Moore [MOORE 72] independently had applied Gordon and Newell's [GORDON 67] work to modeling computer systems. His approach to obtaining an efficient solution for the normalizing constant was based on a partial fraction expansion method. The complexity of this method is O[Ks^2]. Buzen's iterative method is less complex, O[Ks], and also more versatile. Reiser and Kobayashi [REISER 75] have generalized Moore's solution technique for mixed networks (i.e. a mix of both open and closed subchains) and removed several of the previous modeling constraints. They provide a general algorithm based on multiplication of power series, which can be viewed as a multi-dimensional linear filter. In an independent effort Lam [LAM 77B] had extended Moore's solution technique to include nondistinct traffic intensities.

The concept of "local balance" was introduced by Chandy [CHANDY 72]; the previous works by Jackson, Gordon and Newell, and Buzen were based on "global balance." Local balance is a subset of global balance in which one concentrates on the flow through a single queue, rather than through all queues. More specifically it requires equivalent terms on one side of the balance equation to equal those on the other side, instead of the more general solution to the equation.

Chandy has also shown that some other service time distributions and scheduling policies yield the same results as the exponential distribution with a FCFS scheduling policy. Therefore, only the mean service time is required in the product-form solution for these other distributions and corresponding scheduling policies. The four cases that Chandy identifies as having equivalent solutions to the exponential distribution are: (1) exponential service distributions and FCFS scheduling; (2) geometric service distributions whose Laplace transform is rational and a processor sharing scheduling policy; (3) a service distribution whose Laplace transform is rational and a processor sharing scheduling policy; and (4) a service distribution whose Laplace transform is rational and a last-come-first-served-premptive-resume (LCFS) scheduling policy. One way to

understand the relationship between case 1 (previouly the standard) and the others is to realize that a rational Laplace transform inverts to a sum of exponential distributions [COX 55], in other words, a hypo- or hyper-exponential distribution. A hypo-exponential distribution is realizable as a set of serial exponential stages, while a hyper-exponential distribution is realizable as a set of parallel exponential stages.

Another attempt to extend the range of probability distributions of the central server model is due to Baskett and Gomez [BASKET 72]. Using an approach similar to Chandy's [CHANDY 72] "case-3" for the CPU, a service time distribution with a rational Laplace transform and a processor sharing scheduling policy, they introduced the coefficients of variation into their model. For this model they derived the equilibrium state probabilities of the network which are identical to Buzen's, so therefore, all of Buzen's results extend to this variation.

Baskett and Muntz [BASKET 73] extended Chandy's [CHANDY 72] local balance model by incorporating multiple classes of customers. The allowable classes are obtained from the four cases presented by Chandy. Each service center contributes a factor dependent on its class to the product form solution. The equilibrium state probability distribution is presented in two forms, a detailed form which denotes customer class per service center and an aggregate form of total customers per service center, the latter exists only under specific conditions. Due to the properties of local balance the marginal state distributions for open networks are obtained in closed form. The resemblance between these marginal distributions and those of single server systems is striking. The marginal distributions are equivalent to those of an M/M/1 queue (a single server queue with Poisson arrival and exponential service time distribution), and the exception (infinite server, and iid rational Laplace transform) is equivalent to that of an M/G/1 queue (a single server queue with Poisson arrival and general service time distribution).

The more recent results of queueing network theory, using multiple job classes, are applicable to our SPR acrhitecture. The primary limitation is the memory-space complexity closely followed by the time complexity of obtaining solutions. The state-space for these models grows in an exponential manner. Although efficient algorithms exist for these models, the SPR architecture and many other systems have state-spaces too large to be reasonably evaluated by these algorithms, especially when multiple job classes are involved.

C. Approximations

The current queueing network limitations of size and structure have provided the motivation for approximate modeling techniques. The thrust of the approximation techniques has attacked the structure limitation. This thrust has occured because previously most subsystems had a state-space whose size was reasonable to evaluate using existing queueing network techniques, but whose operational structure was not. Therefore, rather than applying an accurate analytic model to an inaccurate descriptive model, approximate analytic models were formulated to approximate an accurate descriptive model. As a result of these approximation techniques some economy on the size limitation has also been realized.

These approximation techniques can be categorized as decomposition or substitution, neither precludes the use of the other. Decomposition separates the system into various pieces, each piece is modelled individually to obtain a constituent model, and then these constituent models are joined together to form the overall system model. To be able to insure some correlation between the individual constituent models and the overall model, some common relations must usually be satisfied. Although each individual constituent model, and usually the overall system, separately satisfy these relations they do not do so in a consistent manner. Therefore, techniques to coordinate consistent interrelations are required. These relations are generally concerned with the flow and capacity aspects of the system.

The basis for this decomposition into individual subsystems stems from the work done by Courtois [COURTO 71]. Courtois investigated the level of coupling between queues and determined that the subsystem selection should be based on this parameter. Queues that were strongly coupled should be decomposed into subsystems such that the coupling between subsystems is weak. The error introduced by this approximation technique is proportional to the degree of coupling.

The other general approximation technique is substitution. The substitution technique is concerned with substituting one form (or model) for another in an existing solution method. For

example, one might substitute a M/G/1 for a M/M/1 queue form in an analytic model to approximate the operation of a M/G/1 queue in the descriptive model. The concept is that the analyst feels the inaccuracies introduced as a result of this approximation are worth the expediency of utilizing existing methods rather than formulating a new, more complex model or method.

Much of the effort to extend the structure limitations have been concerned with the service time probability distribution. An exponential service time distribution is the most prominent. When an exponential distribution is assumed in formulating a model, it contributes to tractable solutions. This is primarily due to two properties of the exponential distribution. First, it may be specified by a single parameter, its mean. Second, it is memoryless, requiring no previous information to determine its future operation.

A wide range of analytic models are based on the exponential service time distribution assumption, while the real systems they are applied to do not possess such service times. In many of these cases when the analytic predictions were compared to actual measurements a reasonable agreement was observed. Although efforts continue to formulate models based on more general service time distributions, the robustness of the exponential distribution should not be dismissed. This robustness has been discussed [BASKET 72A] and investigated [GROSS 75], and as a result some quantification of the expected error is available for some approximate substitution modeling applications.

Decomposition techniques have been applied by others [BROWNE 75], but these were basically specific to a given model or situation. Chandy [CHANDY 75A] introduced the concept of an equivalent queue, analogous to Norton's theorem in electrical networks. This allows one to transform a subsystem of service centers into a single equivalent service center of a queueing network. This resulting composite service center, referred to as the complement, captures the interface between a specific queue and the rest of the network. Chandy has shown that the equilibrium queue length and wait time distributions of the non-reduced service centers are equilvalent to those of the original network, provided local balance [CHANDY 72B] is satisfied by the network. This means that for systems that satisfy local balance, this technique produces exact results. For systems that do not satisfy local balance Chandy [CHANDY 75B] has generalized the above decomposition technique by using an iterative converging flow balance relation to "adjust" the service rate (flow) of the composite queue.

The intent of this decomposition technique was to reduce the complexity of the analysis task. If one were interested in the analysis of a specific service center, then all other service centers could be represented by a single composite queue resulting in a two queue system, which is generally less complex to analyze then the larger original system. Systems whose decomposition results in multi-class composite queues must also consider job ordering of the different classes, or sub-chains. The resultant state-space is generally not decreased and little, if any, economy is realized in the complexity of the analysis. In developing our exact SPR model in chapter III, we have applied this decomposition technique. Since multiple job classes are being dealt with, only marginal economy is achieved and it is still necessary to evaluate an extremely large state-space.

Currently the product form solution [BASKET 75] of queueing network theory has evolved to a structure allowing a general connectivity of a variety of service centers each of which may have a general service time distribution (having a rational Laplace transform), and may also have a number of different scheduling policies and job classes. A service center having a FCFS scheduling policy, however, is constrained to have an exponential service time distribution. In an effort to extend this structure even further Shum [SHUM 77] presented an "extended product form" (EPF) solution method. Based on the fact that each product factor has the form of an M/M/1 marginal distribution, Shum postulated that a reasonable approximation would be to replace each factor with an appropriate M/G/1 marginal distribution form. The basic convolution computations and performance measures of product form queueing network theory are still applicable to obtaining solutions for this approximate model.

Utilizing this approximate method requires an additional constraint on the solution of the initial set of simultaneous equations. Prior to this approximation the solution to these simultaneous equations resulted in "relative" visit frequencies, which were related to the absolute visit frequencies by a multiplicative constant. This multiplicative constant has no effect on the resultant equilibrium state probabilities or the performance measures, since it is cancelled out of the expression for the product form solution. Therefore, the relative visit frequencies are sufficient for the product form solution. The EPF model, on the other hand, requires that the "absolute" visit frequency be used and, therefore, the multiplicative constant must be obtained.

There is no known method for computing the "absolute" visit frequencies. Shum suggests that balancing a flow relation containing the M/G/1 substitution factors could be used to obtain them. Once this relation is formulated a bounded binary search process could be used to satisfy a least square error criterion. A procedure is presented for computing the M/G/1 product factors for a general distribution with different coefficient of variation (C). When compared to other models (machine repairman model, cyclic model, and central server model) with similar corresponding parameters, an error analysis showed that the largest error, as a function of the coefficient of variation, occured in "mid-range," while exact results were obtained for C=1 (exponential) and diminishing errors resulted for large values of C. Shum has indicated that further work is needed to extend the EPF model to include multiple classes, and queue dependent service times. The approximation developed by Shum is an example of the combined use of both substitution (M/G/1 for M/M/1) and decomposition or flow equivalent techniques (used to obtain the absolute visit frequencies). The EPF approximation is not applicable for the SPR architecture principally due to the need to account for multiple job classes.

An approximate solution for queueing networks has been developed by Kobayashi [KOBAYA 74A], using the Kolmogorov diffusion equations (also known as Fokker-Planck equations). By using a "Central Limit-Theorem" argument, Kobayashi has hypothesized that changes in queue length over a large enough time interval approximates a stochastic process with a normal distribution. As a result, this queue length process can be modeled by a Wiener-Levy process (or Brownian motion) with a suitable boundary condition. Equations for the equilibrium state probability distribution for a queue are developed. When these results are compared to the known solution of a M/M/1 queue they are found to be in error. In an effort to reduce this error, these results are modified to conform to this known solution. Using a multi-dimensional diffusion equation this approach is extended to both open and closed networks. Once the the equilibrium state probabilities are obtained they are then substituted back into the product form solution. For the SPR architecture we have already assumed an exponential service process. Therefore, existing exact queueing network theory can be used without the need to determine the probability distributions by using the diffusion approximation.

Using this approximate technique the transient state probability distributions are obtainable. Kobayashi [KOBAYA 74B] derives them for a single-server and a cyclic queueing system. Obtaining the transient solution to general open and closed networks is much more difficult, and no method currently exists.

Decomposition or flow equivalent techniques apply to exact as well as approximate models. In certain situations these composite sub-models can significantly reduce the size of the overall system state-space, therefore, allowing large system models to be evaluated much more efficiently. There are some problems with this technique. One is that the service center or centers that one may desire to investigate must usually be excluded from any of the composite sub-models. Another is if any parameters of the service centers within a sub-model changes a new sub-model must usually be evaluated and "re-aligned" to agree with the overall model consistencies. This re-evaluation process may negate any size economy that may otherwise been realized. Although the decomposition technique is applicable to the SPR architecture it does not reduce the state-space size. Also, when these techniques are applied to approximate models the resultant accuracy is generally variable and unknown. Chandy [CHANDY 75] has indicated a 10% to 20% expected error for his approximation technique. Others [SHUM 76, KOBAYA 74A] have indicated an error exists, usually by example for a few configurations, but do not indicate what one can expect for the general case.

We have mentioned earlier the robustness of the exponential distribution. Realizing that this assumption does result in inaccuracies some studies have been conducted to investigate them as well as errors caused by substituting other service time distributions. Gross [GROSS 75] has investigated the resultant error when a M/M/1 queueing model is used to approximate a M/G/1 queue. He found that the resulting error was proportional to the coefficient of variation. Buzen [BUZEN 74] has investigated the use of a M/G/1 queueing model to approximate a M/G/1/K queue. He found that the largest error occured when the traffic intensity was high or the value of K was low. Buzen [BUZEN 77] also investigated the use of a M/M/1/K queueing model (a single server queue with Poisson arrival and exponential service time distribution, with a queueing limit of K jobs) to approximate a M/G/1/K queue (a single server queue with Poisson arrival and general service time distribution, with a queueing limit of K jobs). His results indicate that

response time performance measures have a greater sensitivity to this approximation than do the utilization or throughput performance measures. Also Buzen indicates that the error is proportional to the coefficient of variation, which independently verifies similar results obained by Gross.

All of these studies have helped to quantify the error that results from these substitution approximations. Therefore, they provide the analyst with some quantification of the error that can be expected when these substitution approximations are applied. In a similar manner we attempt to quantify the error that can be expected from the approximate model presented in chapter IV.

III. Shared Central Server Model

A. The Model

Utilizing the general theory of queueing networks, a model will be constructed for a special class of computer architecture. The model (figure III-1) is a network consisting of a set of independent computing systems (ICS) and a single shared processing resource (SPR). Each ICS is a central server system (figure III-2) consisting of a single central server (CPU) and a number of peripheral processors (PPUs). Each ICS processes a separate class of jobs, while the SPR processes all classes of jobs. Class distinction is the means by which the job flow from each ICS is kept segregated. This model to be developed will be referred to as the Shared Central Server (SCS) Model.

The SPR and the devices within an ICS each represent an intricate subsystem, as do the various communication subsystems interconnecting each of them. The model does not directly incorporate the effects of various strategies that may be utilized within any of the subsystems, such as the effect of different communication protocols. Instead these effects are assumed either to be incorporated into the device processing time or to be insignificant when compared to the overall device processing time.

Specifically, this model consists of R ICSs where the i-th ICS is composed of s_i devices. For the i-th ICS, the CPU is denoted as device (i,1), and devices (i,2) through (i,s_i) are the PPUs. The SPR is designated as device (i,0), or just (0), and has a device count $s_0 = 1$. The network structure can be represented by a vector denoting the number of devices in each ICS, $S = (s_0, \ldots, s_R)$, and the total number of devices in the network is $L = \Sigma\, s_i$. The network is closed, constantly circulating and processing a total of K jobs, which are composed of separate and distinct classes, one for each ICS; while the SPR processes every class using a FCFS scheduling policy. A vector description of the job class allocation is $J = (J_1, \ldots, J_R)$, where J_i is the maximum number of (class i) jobs in the i-th ICS. The network state n is defined as the distribution of the various classes of jobs among all the devices, $n = (n_{1,0}, n_{1,1}, \ldots, n_{1,s_1}, n_{2,0}, \ldots, n_{R,s_R})$, where $n_{i,j}$ is the number of class i jobs both waiting and being processed at device j in the i-th ICS. The network job constraints are

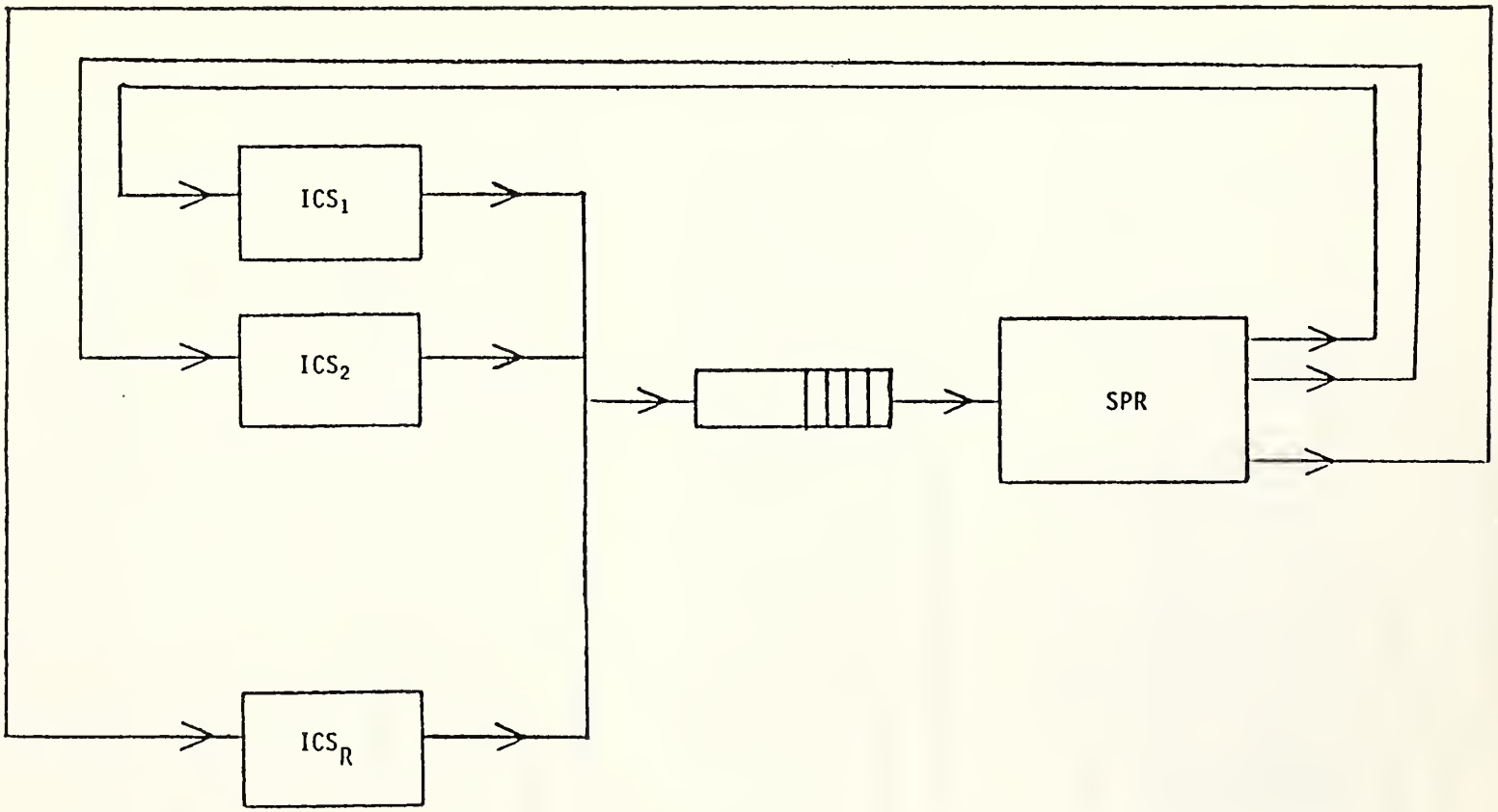

Figure III-1. SCS block diagram. Simplified block diagram of the SCS (Shared Central Server) model showing the separate flow path of each ICS. The box preceding the SPR is an explicit representation of its queue.

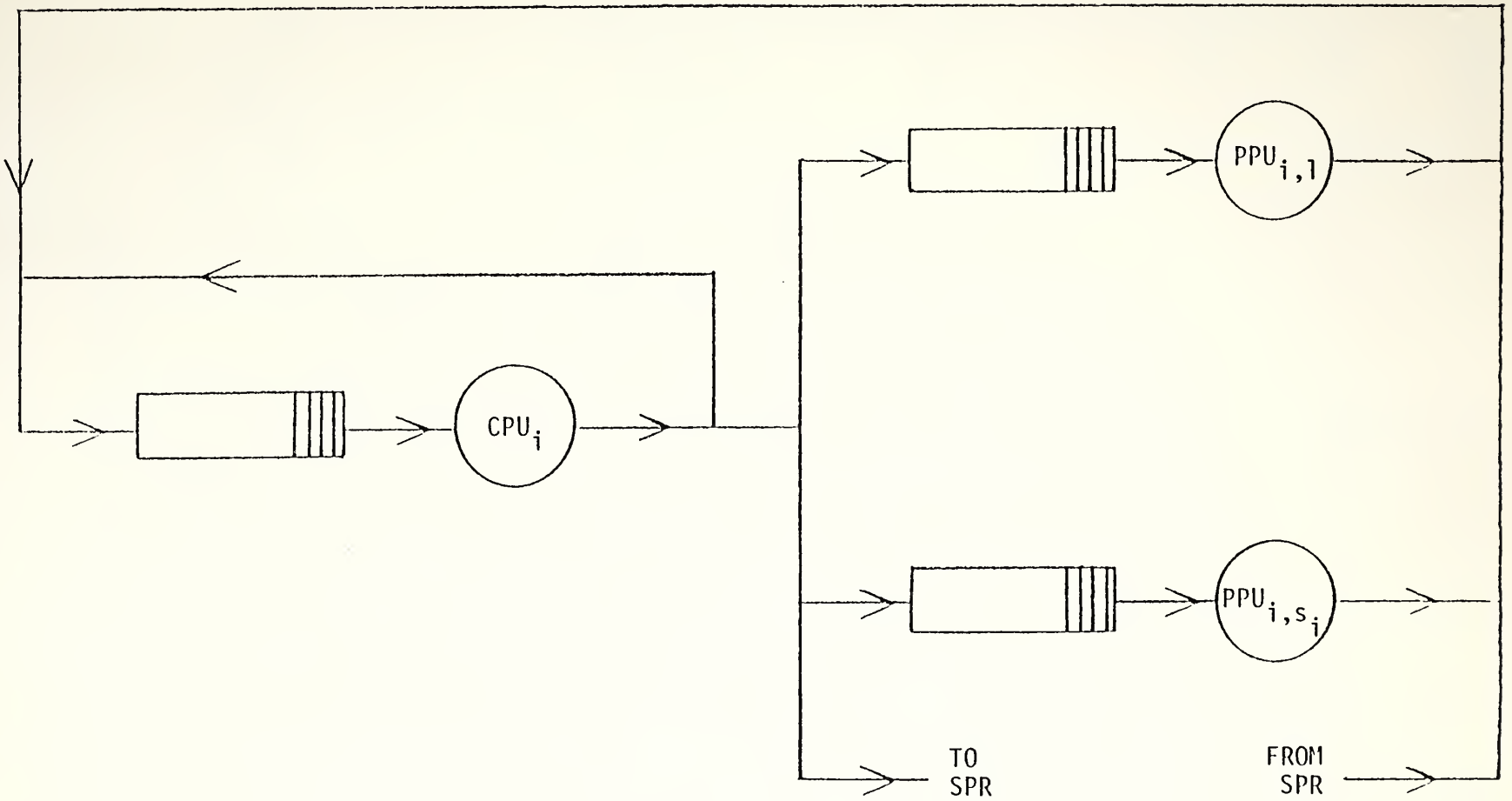

Figure III-2. ICS block diagram. Simplified block diagram of the i-th ICS (Independent Computing System) showing the job flow. Each circle represents a service center (device) and the box preceding it represents its queue.

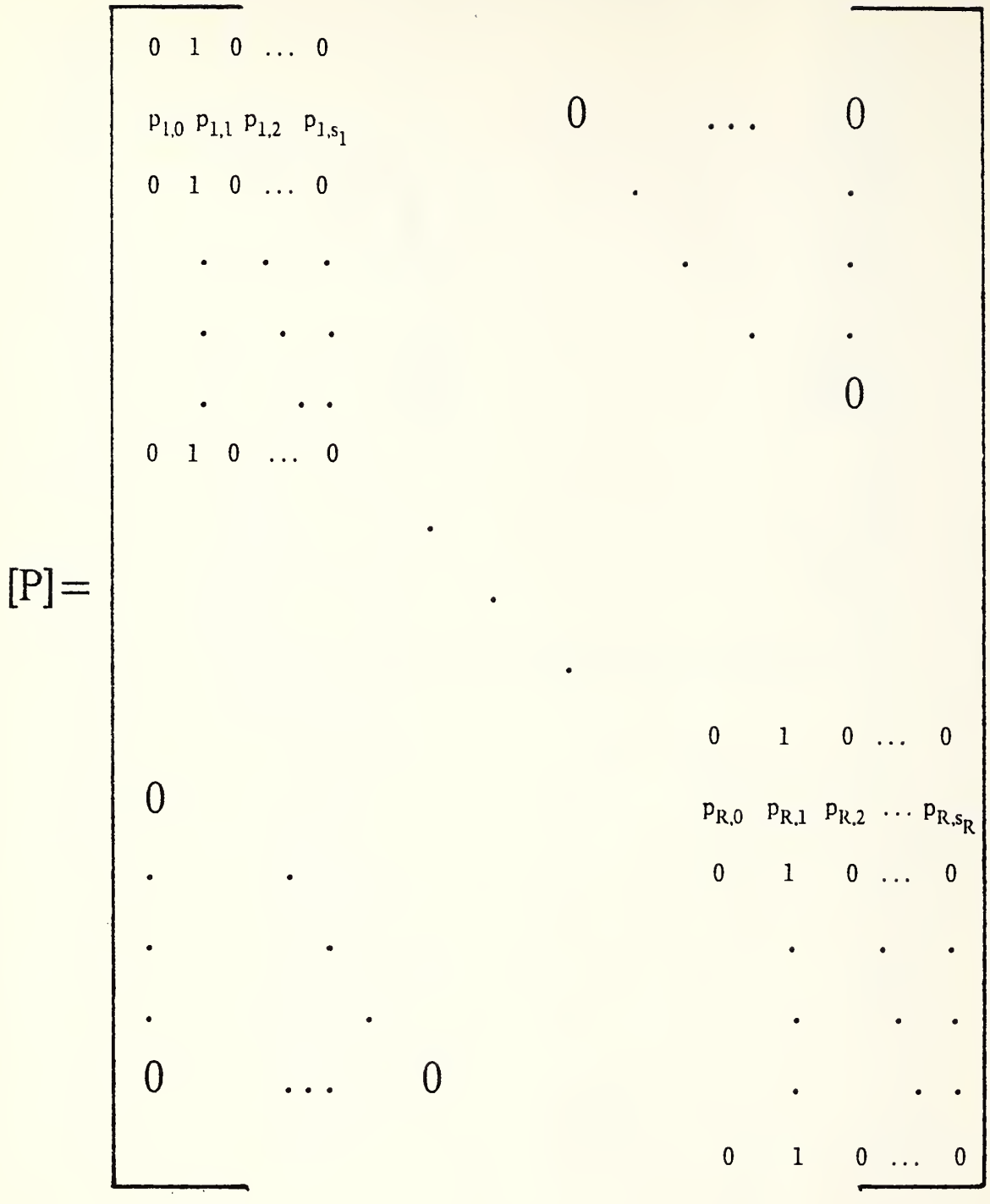

Figure III-3. SCS model transition probability matrix.

(1a) $$n_i = \sum_{j=1}^{s_i} n_{i,j} \leq J_i \quad , \quad i>0$$

(1b) $$N_0 = (n_{1,0} , \ldots , n_{R,0})$$

(1c) $$\sum_{i=1}^{R} J_i = \sum_{i=1}^{R} \sum_{j=0}^{s_i} n_{i,j} = K \quad , \text{ and}$$

(1d) $$n_i = \begin{cases} \sum_{j=1}^{s_i} n_{i,j} & , \quad i>0 \\ \sum_{j=1}^{R} n_{j,0} & , \quad i=0 \end{cases}$$

At each device jobs are processed on a FCFS basis with an exponential service time distribution that is independent and identically distributed (iid), and not dependent on the number of jobs waiting for service. The j-th device of the i-th ICS is characterized by its mean processing service rate $u_{i,j}$, where $u_{i,0}=u_0$ for all i. The job flow (as indicated in figures III-1 and III-2) is from the i-th CPU to one of the devices within the i-th ICS or to the SPR with a constant known probability, $0<p_{i,1;i,j}<1$, where $0\leq j\leq s_i$. Note that the jobs do not change class as they transit between devices. After being processed by a PPU or the SPR, the job returns to the i-th CPU with probability $p_{i,j;i,1}=1$. All other transition probabilities are equal to zero. The transition probability matrix for the SCS model is shown in figure III-3. The $p_{i,1;i,j}$'s are the only transition probabilities that are not 0 or 1 and, therefore, they are the only transition probabilities that need to be specified by a variable. Based on this we may shorten the subscript notation for these transition probabilities to $p_{i,j}$. The SCS transition probability matrix can be conceptualized as a matrix whose diagonal is a set of sub-matrices and all non-diagonal elements are zero. Each sub-matrix is an (s_i+1) x (s_i+1) matrix, with a single non-zero row and column.

Within the model the job flow is assumed to occur in zero time, although in an actual system a finite amount of time is required. The communication subsystem for these architectures generally has sufficient excess bandwidth to prevent it from becoming a bottleneck [WATSON 80, THORTO 80]. If the communication time is significant compared to the device processing time, then the device processing time can be increased to account for it. So, this modeling assumption is reasonable.

With this "micro" information one can then compute the relative load factor [GIAMMO 76], $x_{i,j}$, for each device by solving a set of simultaneous equations as follows:

(2a) $$e_{i,j} = \sum_{m=1}^{R} \sum_{k=0}^{s_i} e_{m,k}\, p_{m,k;i,j} \quad , \text{ and}$$

(2b) $$x_{i,j} = \frac{e_{i,j}}{u_{i,j}} \quad ,$$

where $e_{i,j}$ is the relative visit frequency to the j-th device in the i-th ICS by a class i job. For the SCS model the relative visit frequencies can be found in terms of $e_{i,1}$ by substituting the transition probabilities of figure III-3 into (2a). Choosing $e_{i,1} = u_{i,1}$ results in the following (See Appendix A Section 3):

$$e_{i,j} = \begin{cases} u_{i,1} & , \quad j=1 \quad \text{(CPU)} \\ p_{i,j}\, u_{i,1} & , \quad j \neq 1 \quad \text{(not CPU)} \end{cases} \quad .$$

Hence, $x_{i,1} = 1$ by choice. Using the relative load factors, the state equilibrium probabilities can be computed by utilizing the product form solution [BASKET 75] (also see Appendix A, sections 3 and 4)

(3) $$P(n) = \frac{1}{G(J)}\, f_0(N_0) \prod_{z=1}^{L-1} f_z(n_z) \quad ,$$

where z is a mapping from the double device indices (i,j) to a single index (somewhat analogous to the mapping of a Fortran two dimensional array into a linear address space), such that

$$z = \begin{cases} 0 & , \quad j=0 \quad \forall\, i \\ j & , \quad j>0 \quad i=1 \\ j + \sum_{m=1}^{i-1} s_m & , \quad j>0 \quad i>1 \end{cases} \quad \text{, and} \tag{4}$$

$$L = \sum_{i=0}^{R} s_i \quad .$$

The product factor is (note, the subscript order is reversed from Appendix A, Section 4)

$$f_0(N_0) = n_0 ! \prod_{i=1}^{R} \frac{x_{i,0}^{\,n_{i,0}}}{n_{i,o} !} \quad , \tag{5a}$$

$$f_z(n_z) = x_z^{\,n_z} \quad , \quad z>0 \quad . \tag{5b}$$

The normalization constant is

$$G(J) = \sum_{n} f_0(N_0) \prod_{z=1}^{L-1} f_z(n_z) \quad ,$$

and the sum is over the entire state space n, which is constrained as specified in (1).

A computationally efficient method of computing the normalization constant was first presented by Buzen [BUZEN 71], for a single class of jobs (see appendix A). Others have since presented a generalization of this iterative method for multiple job classes [MUNTZ 74, GIAMMO 76, SHUM 77]. The efficient generalized computation method requires evaluation of an auxiliary function. This auxiliary function has been established as an aid to explicitly present the recursive structure of the normalization constant (Appendix A, sections 3 and 4). This auxiliary function, for our model, is

$$g(M;z) = g(M;z\text{-}1) + x_z \; g(M\text{-}d_i;z) \qquad 1 \leq z \leq L\text{-}1$$

(6)
$$g(M;0) = \sum_{r=1}^{R} x_{r,0} \; f_0(M\text{-}d_r) = |M|! \prod_{r=1}^{R} \frac{x_{r,0}^{m_r}}{m_r!} \quad ,$$

where

$G(J) = g(J;L)$,

$g(\underline{0};z) = 1$, $1 \leq z \leq L\text{-}1$

$M = (m_1, \ldots, m_R)$ is a dummy counting vector that may range over the job allocation vector, $\ni \; 0 \leq m_i \leq J_i$, $||M|| = \prod_{i=1}^{R} (m_i+1)$, and $|M| = \sum_{i=1}^{R} m_i$,

$d_i = (b_1, \ldots, b_R)$ a unit vector, $\ni \; b_r = \begin{cases} 0 & r \neq i \\ 1 & r = i \end{cases}$, $r = 1, \ldots, R$,

$f_0(M) = \sum_{r=1}^{R} x_{r,0} \; f_0(M\text{-}d_r)$, with $f_0(\underline{0}) = 1$,

and z is the mapping of the dual indices into a single index as defined in (4).

To apply this method directly would require retaining all $||M||$ values of $g(M;z)$ for at least a single z. One may conceptualize $g(M;z)$ as an $||M||$ by z matrix and therefore, the retention of a single z set of values would consist of a column containing $||M||$ elements. The resultant minimum storage for $g(J;z)$ is

(7)
$$\prod_{i=1}^{R} (J_i+1) \quad .$$

The total number of operations is on the order of

$$(8) \qquad 2(R+L\text{-}1)\prod_{i=1}^{R}(J_i+1) \quad ,$$

where L is the total number of devices in the network as defined in (4). For example, consider a network consisting of R = 6, K = 30 allocated as J = (5,5,5,5,5,5) and S = (1,3,3,3,3,3,3); which is 6 ICSs (i.e. 6 job classes) each comprising 3 devices and processing 30 jobs allocated as a maximum of 5 per ICS. This example would require minimum storage of $6^6 = 46{,}656$ words and approximately 2,239,488 operations. As the number of ICSs, R, or jobs, K and J, increase the storage requirements and operation count increase exponentially, which can be seen by inspecting (7) and (8).

Starting with the general solution and utilizing the specific structure of the SCS shall allow us to formulate a more efficient procedure for computing the normalization constant. The general queueing network solution for the normalization constant of (3) is

$$(9a) \qquad G(J) = \sum_{n} f_0(N_0)\prod_{z=1}^{L-1} f_z(n_z)$$

$$(9b) \qquad = \sum_{\substack{L-1 \\ \sum n_z = K \\ z=0}} f_0(N_0)\prod_{z=1}^{L-1} f_z(n_z)$$

$$(9c) \qquad = \sum_{\substack{L-1 \\ \sum n_z \le K \\ z=1}} \left\{ f_0(N_0)\prod_{z=1}^{L-1} f_z(n_z) \right\}$$

$$(9d) \qquad = \sum_{\substack{s_i \\ \sum n_{i,j} \le J_i \ , \ \forall i \\ j=1}} \left\{ f_0(N_0)\prod_{i=1}^{R}\prod_{j=1}^{s_i} f_{i,j}(n_{i,j}) \right\}.$$

Noting the structure of the SCS we can factor out those devices which service only the R-th job class, resulting in

$$G(J) = \sum_{\substack{s_i \\ \sum_{j=1} n_{i,j} \le J_i,\ \forall i \le R-1}} \left\{ \prod_{i=1}^{R-1} \prod_{j=1}^{s_i} f_{i,j}(n_{i,j}) \left[\sum_{\substack{s_R \\ \sum_{k=1} n_{R,k} \le J_R}} f_0(N_0) \prod_{k=1}^{s_R} f_{R,k}(n_{R,k}) \right] \right\} . \tag{10}$$

Concentrating on the inner-most factor of (10), and substituting (5) yields

$$\sum_{\substack{s_R \\ \sum_{k=1} n_{R,k} \le J_R}} \left[\left\{ n_0! \prod_{r=1}^{R} \frac{x_{r,0}^{n_{r,0}}}{n_{r,0}!} \right\} \prod_{k=1}^{s_R} x_{R,k}^{n_{R,k}} \right] . \tag{11}$$

Rearranging and further factoring (11) results in

$$\prod_{r=1}^{R-1} \frac{x_{r,0}^{n_{r,0}}}{n_{r,0}!} \sum_{\substack{s_R \\ \sum_{k=1} n_{R,k} \le J_R}} \left[\left\{ n_0! \frac{x_{R,0}^{n_{R,0}}}{n_{R,0}!} \right\} \prod_{k=1}^{s_R} x_{R,k}^{n_{R,k}} \right]$$

$$= \left(\prod_{r=1}^{R-1} \frac{x_{r,0}^{n_{r,0}}}{n_{r,0}!} \right) \left[\sum_{m_R=0}^{J_R} \sum_{\substack{s_R \\ \sum_{k=1} n_{R,k} = m_R}} \left\{ n_0! \frac{x_{R,0}^{n_{R,0}}}{n_{R,0}!} \right\} \prod_{k=1}^{s_R} x_{R,k}^{n_{R,k}} \right] .$$

Noting that $n_{r,0} = J_r - m_r$ further yields

$$\left(\prod_{r=1}^{R-1} \frac{x_{r,0}^{n_{r,0}}}{n_{r,0}!} \right) \left[\sum_{m_R=0}^{J_R} \left(\left\{ n_0! \frac{x_{R,0}^{n_{R,0}}}{n_{R,0}!} \right\} \sum_{\substack{s_R \\ \sum_{k=1} n_{R,k} = m_R}} \prod_{k=1}^{s_R} x_{R,k}^{n_{R,k}} \right) \right] . \tag{12}$$

The inner-most factor of (12) is recognized as a recursive function of the form

$$\sum_{\substack{\sum_{k=1}^{s_R} n_{R,k}=m_R}} \prod_{k=1}^{s_R} x_{R,k}^{n_{R,k}} = \sum_{\substack{\sum_{k=1}^{s_R-1} n_{R,k}=m_R}} \prod_{k=1}^{s_R-1} x_{R,k}^{n_{R,k}} + x_{R,s_R} \sum_{\substack{\sum_{k=1}^{s_R} n_{R,k}=m_R-1}} \prod_{k=1}^{s_R} x_{R,k}^{n_{R,k}} \quad .$$

This can be expressed on a term by term basis by a family of auxiliary functions, one for each ICS, in a form similar to (6) by letting:

$$g_r(m_r) = g_r(m_r;s_r) = g_r(m_r;s_r-1) + x_{r,s_r} g_r(m_r-1;s_r) \quad ,$$

where

$$g_r(0;s_r) = 1 \quad , \text{ and}$$

$$g_r(m_r;1) = x_{r,1}^{m_r} \quad .$$

Therefore, (12) becomes

$$(13) \qquad \left(\prod_{r=1}^{R-1} \frac{x_{r,0}^{n_{r,0}}}{n_{r,0}!} \right) \left[\sum_{m_R=0} n_0! \frac{x_{R,0}^{n_{R,0}}}{n_{R,0}!} g_R(m_R) \right] \quad .$$

Substituting (13) into (10) results in

$$G(J) = \sum_{\substack{\sum_{j=1}^{s_i} n_{i,j} \le J_i, \forall i \le R-1}} \left\{ \prod_{i=1}^{R-1} \prod_{k=1}^{s_i} f_{i,k}(n_{i,k}) \frac{x_{i,0}^{n_{i,0}}}{n_{i,0}!} \left[\sum_{m_R=0}^{J_R} n_0! \frac{x_{R,0}^{n_{R,0}}}{n_{R,0}!} g_R(m_R) \right] \right\}$$

$$= \sum_{\substack{\sum_{j=1}^{s_i} n_{i,j} \le J_i, \forall i \le R-1}} \left\{ \prod_{i=1}^{R-1} \frac{x_{i,0}^{n_{i,0}}}{n_{i,0}!} \prod_{k=1}^{s_i} f_{i,k}(n_{i,k}) \left[\sum_{m_R=0}^{J_R} n_0! \frac{x_{R,0}^{n_{R,0}}}{n_{R,0}!} g_R(m_R) \right] \right\} .$$

Repeating the above partitioning and factoring process results in the following

$$G(J) = \sum_{m_1=0}^{J_1} \left\{ \frac{x_{1,0}^{n_{1,0}}}{n_{1,0}!}\, g_1(m_1) \left\{ \ldots \left\{ \sum_{m_{R-1}=0}^{J_{R-1}} \frac{x_{R-1,0}^{n_{R-1,0}}}{n_{R-1,0}!}\, g_{R-1}(m_{R-1}) \right. \right. \right. \tag{14}$$

$$\left. \left. \left. \left\{ \sum_{m_R=0}^{J_R} n_0! \frac{x_{R,0}^{n_{R,0}}}{n_{R,0}!}\, g_R(m_R) \right\} \right\} \ldots \right\} \right\} .$$

Letting

$$h(i;m_i) = \frac{x_{i,0}^{n_{i,0}}}{n_{i,0}!}\, g_i(m_i) \quad ,$$

results in

$$G(J) = \sum_{m_1=0}^{J_1} h(1;m_1) \left[\ldots \left[\sum_{m_{R-1}=0}^{J_{R-1}} h(R-1;m_{R-1}) \left[\sum_{m_R=0}^{J_R} h(R;m_R)\, n_0! \right] \right] \ldots \right] \tag{15}$$

$$= \sum_{m_1=0}^{J_1} \ldots \sum_{m_R=0}^{J_R} \left[n_0! \prod_{i=1}^{R} h(i;m_i) \right] \tag{16}$$

$$= \sum_{n_0=0}^{K} n_0! \left[\sum_{\substack{R \\ \Sigma n_{i,0}=n_0 \\ i=1}} \prod_{i=1}^{R} h(i;J_i-n_{i,0}) \right] \tag{17}$$

$$= \sum_{m_1=0}^{J_1} \ldots \sum_{m_R=0}^{J_R} f_0(J-M) \prod_{i=1}^{R} g_i(m_i) \quad , \text{ and} \tag{18}$$

$$f_0(M) = \sum_{i=1}^{R} x_{i,0}\, f_0(M-d_i)$$

Using the form of (18) a recursive relation can be formulated similar to (6) as follows

$$G(J) = \sum_{m_1=0}^{J_1} \cdots \sum_{m_R=0}^{J_R} \{ f_0(J-M) \prod_{i=1}^{R} g_i(m_i) \}$$

$$= \sum_{m_1=0}^{J_1} \cdots \sum_{m_R=0}^{J_R} \{ [\sum_{r=1}^{R} x_{r,0} f_0(J-d_r-M)] \prod_{i=1}^{R} g_i(m_i) \}$$

$$= f_0(0) \prod_{i=1}^{R} g_i(J_i) + \sum_{r=1}^{R} x_{r,0} \{ \sum_{m_1=0}^{J_1} \cdots \sum_{m_r=0}^{J_r-1} \cdots \sum_{m_R=0}^{J_R} f_0(J-d_r-M) \prod_{i=1}^{R} g_i(m_i) \}.$$

Noting that

$$G(J-d_r) = \sum_{m_1=0}^{J_1} \cdots \sum_{m_r=0}^{J_r-1} \cdots \sum_{m_R=0}^{J_R} f_0(J-d_r-M) \prod_{i=1}^{R} g_i(m_i) \quad , \text{ and}$$

$$f_0(\underline{0}) = 1 \quad ,$$

yields a recursive relation for the normalization constant as

$$G(J) = \prod_{i=1}^{R} g_i(J_i) + \sum_{r=1}^{R} x_{r,0} G(J-d_r), \tag{19}$$

where $G(0)=1$.

The above normalization constant is computed as the convolution of the product factors for the SPR and the auxiliary "g" functions for each ICS over the range of the job class allocation vector. This means that due to the structure of the SCS we are able to represent each ICS by a

single composite "g" function rather than by a set of product factors, one for each device. The "g" functions represent the convolution of the product factors for each device in an ICS, and can be computed efficiently.

The state equilibrium probability is obtained by solving for G(J), by using any of (15) through (19), which can then be substituted into (3) to yield

$$P(n) = \frac{1}{G(J)} f_0(N_0) \prod_{i=1}^{R} \prod_{j=1}^{s_i} f_{i,j}(n_{i,j}) \quad . \tag{20}$$

B. Performance Measures

Having established the basic relationships to compute the state probabilities, we shall now utilize them to form relationships for some performance measures. We shall temporarily exclude the SPR from the following. The threshold equilibrium queue length probability distribution, which is the marginal probability that device (i,j) is serving k or more jobs, is

$$\begin{aligned} P[n_{i,j} \geq k] &= \sum_{n \ni n_{i,j} \geq k} P(n) \\ &= \frac{1}{G(J)} \sum_{n \ni n_{i,j} \geq k} f_0(N_0) \prod_{r=1}^{R} \prod_{t=1}^{s_r} f_{r,t}(n_{r,t}) \quad . \end{aligned} \tag{21}$$

From (5) the expression for $f_{r,t}(n_{r,t})$ is $x_{r,t}^{n_{r,t}}$ and, therefore, when $n_{i,j} \geq k$ a factor of $x_{i,j}^{k}$ can be extracted from (21). This extraction changes the job class allocation vector over which sums are taken, from $J=(J_1, \ldots, J_R)$ to $J'=(J_1, \ldots, J_i-k, \ldots, J_R)$ and also causes a corresponding change in the state-space from n to n'. Applying these transformations to (21) results in

$$P[n_{i,j} \geq k] = \frac{1}{G(J)} x_{i,j}^{k} \sum_{n'} f_0(N_0) \prod_{r=1}^{R} \prod_{t=1}^{s_r} f_{r,t}(n_{r,t}) \quad .$$

Noting the similarity of the summation portion of the above expression to (9) results in

$$P[n_{i,j} \geq k] = \frac{G(J')}{G(J)} x_{i,j}^{k}$$

(22)

$$= \frac{G(J-kd_i)}{G(J)} x_{i,j}^{k} \quad , \text{ for } i,j \neq 0 \quad .$$

If the marginal probability of device (i,j) is desired, it may be expressed as

$$P[n_{i,j}=k] = P[n_{i,j} \geq k] - P[n_{i,j} \geq k+1]$$

$$= \frac{1}{G(J)} \left[x_{i,j}^{k} G(J-kd_i) - x_{i,j}^{k+1} G(J-(k+1)d_i) \right]$$

(23)

$$= \frac{x_{i,j}^{k}}{G(J)} \left[G(J-kd_i) - x_{i,j} G(J-(k+1)d_i) \right] \quad , \text{ for } i,j \neq 0 \quad .$$

Of more interest than these probabilities are the performance measures of device (i,j), such as the busy probability, $A_{i,j}$, the mean queue length, $Q_{i,j}$, and average throughput, $T_{i,j}$. The device busy probability is obtained from the threshold marginal probability by noting that

(24) $$A_{i,j} = P[n_{i,j} \geq 1] = \frac{G(J-d_i)}{G(J)} x_{i,j} \quad , \text{ for } i,j \neq 0 \quad .$$

The mean queue length of device (i,j) is by definition

$$Q_{i,j} = \sum_{k=1}^{J_i} k \, P[n_{i,j}=k]$$

$$= \sum_{k=1}^{J_i} k \left\{ P[n_{i,j} \geq k] - P[n_{i,j} \geq k+1] \right\}$$

$$= \sum_{k=1}^{J_i} k\ P[n_{i,j} \geq k] \ - \sum_{k=1}^{J_i} k\ P[n_{i,j} \geq k+1]$$

$$= \sum_{k=1}^{J_i} k\ P[n_{i,j} \geq k] \ - \left\{ \sum_{k=2}^{J_i} (k-1)\ P[n_{i,j} \geq k] \ + J_i\, P[n_{i,j} \geq J_i+1] \right\} .$$

Noting that $P[n_{i,j} > J_i] = 0$ results in

$$Q_{i,j} = P[n_{i,j} \geq 1] \ + \sum_{k=2}^{J_i} k\ P[n_{i,j} \geq k] \ - \left\{ \sum_{k=2}^{J_i} k\ P[n_{i,j} \geq k] \ - \sum_{k=2}^{J_i} P[n_{i,j} \geq k] \right\}$$

(25)

$$= \sum_{k=1}^{J_i} P[n_{i,j} \geq k] \quad , \quad \text{for } i,j \neq 0 \quad .$$

Substituting (22) into (25) yields

(26)
$$Q_{i,j} = \frac{1}{G(J)} \sum_{k=1}^{J_i} x_{i,j}^{\,k}\, G(J-kd_i) \quad , \quad \text{for } i,j \neq 0 \quad .$$

The device throughput when the service rate is independent of the queue size is defined as

$$T_{i,j} = \sum_{k=1}^{J_i} u_{i,j}\ P[n_{i,j} = k]$$

$$= u_{i,j} \sum_{k=1}^{J_i} P[n_{i,j} = k]$$

(27)

$$= u_{i,j} \ \{ 1 - P[n_{i,j} = 0] \ \}$$

$$= u_{i,j}\ A_{i,j}$$

$$= \frac{G(J-d_i)}{G(J)} \; e_{i,j} \quad , \quad \text{for } i,j \neq 0 \; .$$

The i-th ICS throughput for service rates independent of queue size is defined to be that of its CPU, which is

(28) $$T_i = \frac{G(J-d_i)}{G(J)} \; e_{i,1} \; .$$

It should be noted that although this measure is referred to as throughput, it may more properly be thought of as effective processing rate or departure rate, as can be seen from its definition in (27).

Similar measures for the SPR will now be derived. The SPR busy probability is

(29) $$A_0 = P[n_0 \geq 1] = 1 - P[n_0 = 0]$$

From (1), it can be seen that when $n_0 = 0$, $n_{i,0} = 0$ and $\sum_{j=1}^{s_i} n_{i,j} = J_i$ for $i = 1, \ldots, R$. From (5), when $n_0 = 0$ it can be seen that $f_0(\underline{0}) = 1$. Substituting this into (20) yields

$$P[n_0 = 0] = \frac{1}{G(J)} \sum_{\substack{s_i \\ \sum_{j=1} n_{i,j} = J_i \, , \; \forall i}} \prod_{i=1}^{R} \prod_{j=1}^{s_i} f_{i,j}(n_{i,j}) \; .$$

Repeating the same partitioning and factoring process used to obtain (14) results in

$$P[n_0 = 0] = \frac{1}{G(J)} \prod_{i=1}^{R} g_i(J_i) \; .$$

Substituting this into (29) yields

(30) $$A_0 = P[n_0 \geq 1] = 1 - \prod_{i=1}^{R} \frac{g_i(J_i)}{G(J)} \quad .$$

To obtain the mean queue length of the SPR its aggregate marginal probability, independent of class, must first be computed. This may be expressed as

$$P[n_0 = k] = \sum_{n \ni \sum_{i=1}^{R} n_{i,0} = k} P(n)$$

$$= \frac{1}{G(J)} \sum_{n \ni \sum_{i=1}^{R} n_{i,0} = k} f_0(N_0) \prod_{i=1}^{R} \prod_{j=1}^{s_i} f_{i,j}(n_{i,j})$$

$$= \frac{1}{G(J)} \sum_{n \ni \sum_{i=1}^{R} n_{i,0} = k} \left\{ \left[k! \prod_{r=1}^{R} \frac{x_{r,0}^{n_{r,0}}}{n_{r,0}!} \right] \left[\prod_{i=1}^{R} \prod_{j=1}^{s_i} x_{i,j}^{n_{i,j}} \right] \right\}$$

$$= \frac{k!}{G(J)} \sum_{n \ni \sum_{i=1}^{R} n_{i,0} = k} \prod_{r=1}^{R} \left\{ \left[\frac{x_{r,0}^{n_{r,0}}}{n_{r,0}!} \right] \prod_{j=1}^{s_r} x_{r,j}^{n_{r,j}} \right\}$$

$$= \frac{k!}{G(J)} \sum_{\sum_{i=1}^{R} n_{i,0} = k} \left[\sum_{n-N_0} \prod_{r=1}^{R} \frac{x_{r,0}^{n_{r,0}}}{n_{r,0}!} \left\{ \prod_{j=1}^{s_r} x_{r,j}^{n_{r,j}} \right\} \right]$$

$$= \frac{k!}{G(J)} \sum_{\sum_{i=1}^{R} n_{i,0} = k} \left\{ \prod_{r=1}^{R} \frac{x_{r,0}^{n_{r,0}}}{n_{r,0}!} \left[\sum_{\sum_{j=1}^{s_r} n_{r,j} = J_r - n_{r,0}} \prod_{j=1}^{s_r} x_{r,j}^{n_{r,j}} \right] \right\}.$$

Substituting, into the above, as in (13) yields an expression for the aggregate marginal probability for the SPR as

$$(31) \qquad P[n_0=k] = \frac{k!}{G(J)} \sum_{\substack{R \\ \Sigma n_{i,0}=k \\ i=1}} \left[\prod_{r=1}^{R} \frac{x_{r,0}^{n_{r,0}}}{n_{r,0}!} \; g_r(J_r-n_{r,0}) \right]$$

$$= \frac{k!}{G(J)} \sum_{\substack{R \\ \Sigma n_{i,0}=k \\ i=1}} \prod_{r=1}^{R} h(r;J_r-n_{r,0}) \; .$$

Forming the defining equation for the SPR mean queue length and then substituting (31) results in

$$Q_0 = \sum_{k=1}^{K} k \; P[n_0=k]$$

$$(32) \qquad = \sum_{k=1}^{K} k \left\{ \frac{k!}{G(J)} \sum_{\substack{R \\ \Sigma n_{i,0}=k \\ i=1 \\ \ni \; n_{i,0} \le J_i}} \prod_{r=1}^{R} h(r;J_r-n_{r,0}) \right\}.$$

Also of interest is the mean queue length by job class. The SPR is the only device that processes multiple job classes and is the only device where this performance measure differs from the aggregate mean queue length. The equation for the SPR class marginal probability, the probability of k class r jobs at the SPR, is

$$P[n_{r,0}=k] = \sum_{\substack{n \\ \ni \; n_{r,0}=k}} P(n) \qquad , \qquad \text{for} \quad 0 \le k \le J_r$$

$$= \frac{1}{G(J)} \sum_{n_{1,0}=0}^{J_1} \cdots \sum_{n_{r,0}=k}^{k} \cdots \sum_{n_{R,0}=0}^{J_R} \left\{ n_0 ! \prod_{i=1}^{R} h(i;J_i-n_{i,0}) \right\} \quad . \tag{33}$$

From this the mean queue length for a class r job is defined as

$$Q_{r,0} = \sum_{k=1}^{J_r} k \, P[n_{r,0}=k]$$

$$= \sum_{k=1}^{J_r} k \left[\frac{1}{G(J)} \sum_{n_{1,0}=0}^{J_1} \cdots \sum_{n_{r,0}=k}^{k} \cdots \sum_{n_{R,0}=0}^{J_R} \left\{ n_0 ! \prod_{i=1}^{R} h(i;J_i-n_{i,0}) \right\} \right] \tag{34}$$

$$= \frac{1}{G(J)} \sum_{n_{1,0}=0}^{J_1} \cdots \sum_{n_{r,0}=1}^{J_r} \cdots \sum_{n_{R,0}=0}^{J_R} \left\{ n_0 ! \, n_{r,0} \prod_{i=1}^{R} h(i;J_i-n_{i,0}) \right\}.$$

Forming the defining equation for the throughput of the SPR and then substituting (30) results in

$$T_0 = u_o A_0$$

$$= u_o \left\{ 1 - \frac{\prod_{i=1}^{R} g_i(J_i)}{G(J)} \right\} \quad . \tag{35}$$

We have derived the standard queueing network probabilities and performance measures for the SCS model, which are recapitulated here:

The normalization constant

$$G(J) = \sum_{n_{1,0}=0}^{J_1} \cdots \sum_{n_{R,0}=0}^{J_R} \left\{ f_0(N_0) \prod_{i=1}^{R} g_i(J_i - n_{i,0}) \right\} \quad , \text{ where} \tag{36}$$

$$f_0(N_0) = \sum_{i=1}^{R} x_{i,0} \, f_0(N_0 - d_i) \quad . \tag{37}$$

The device busy probability

$$A_{i,j} = \frac{G(J-d_i)}{G(J)} \, x_{i,j} \quad , \text{ and} \tag{38}$$

$$A_0 = 1 - \frac{\prod_{i=1}^{R} g_i(J_i)}{G(J)} \, . \tag{39}$$

The mean queue length

$$Q_{i,j} = \frac{1}{G(J)} \sum_{k=1}^{J_i} x_{i,j}^{k} \, G(J - k d_i) \quad , \text{ for } i,j \neq 0 \, , \tag{40}$$

$$Q_0 = \frac{1}{G(J)} \sum_{k=1}^{K} k! \, k \left\{ \sum_{\substack{R \\ \sum_{i=1} n_{i,0} = k}} \prod_{r=1}^{R} h(r; J_r - n_{r,0}) \right\}, \text{ and} \tag{41}$$

$$Q_{r,0} = \frac{1}{G(J)} \sum_{n_{1,0}=0}^{J_1} \cdots \sum_{n_{r,0}=1}^{J_r} \cdots \sum_{n_{R,0}=0}^{J_R} \left\{ n_0! \, n_{r,0} \prod_{i=1}^{R} h(i; J_i - n_{i,0}) \right\} . \tag{42}$$

The device throughput

$$T_{i,j} = u_{i,j} A_{i,j} = \frac{G(J-d_i)}{G(J)} e_{i,j} \quad , \quad \text{for } i,j \neq 0 \quad , \text{ and} \tag{43}$$

$$T_0 = u_0 A_0 = u_0 \left\{ 1 - \frac{\prod_{i=1}^{R} g_i(J_i)}{G(J)} \right\} \quad . \tag{44}$$

The system (class) throughput

$$T_i = \frac{G(J-d_i)}{G(J)} e_{i,1} \quad . \tag{45}$$

C. Computational Algorithms

As discussed previously the general iterative procedures for computation of the normalization constant and other performance measures require substantial memory space. In an effort to reduce the memory required without significantly increasing the computations we have reformulated the network expressions utilizing the structure of the SCS model. In addition, we were able to derive expressions for mean queue length, which in general are not available in the literature. Current iterative algorithms to evaluate our expressions require processing time and memory-space that grows exponentially. We will present algorithms to evaluate our expressions which use a minimal amount of memory and require the same order of processing time as do the existing iterative procedures.

Examining the expressions for our performance measures it can be seen that the computations are all very similar, requiring the sum over a restricted state space. Two forms of this computation are represented by (15) and (17), which we shall call the sum-of-products (SOP) expansion and the factorial (FAC) expansion, respectively. Each form has its advantages and disadvantages.

The SOP expansion minimizes the number of multiplications, but places a burden on the factorial computation since the value is not monotonically changing. The FAC expansion simplifies the factorial computation, but requires generation of all states in a restricted sub-space. We will present efficient algorithms for both evaluation forms and for the generation of a restricted sub-space.

The SOP algorithm requires an efficient method to evaluate the factorial, $n_0!$, in the inner-most product term of (15). This value is a function of all the indices and, therefore, is continually changing during the evaluation of (15). If the value of n_0 were monotonically increasing then an efficient method to compute the next factorial value n_0, based on its previous value, is the well known recursion

$$n_0! = (n_0\text{-}1)!\ n_0 \quad .$$

In our case the value of n_0 varies in a cycle which first monotonically increases and then abruptly decreases. This decrease occurs at well defined points; when any product term "sum-limit" is reached. By keeping track of the last value to be factorialized and its factorial value for each product term, the above efficient method may still be applied.

The SOP expansion is of the following form (note: $m_i = J_i - n_{i,0}$):

$$G(J) = \sum_{n_{1,0}=0}^{J_1} h(1;J_1\text{-}n_{1,0}) \left[\ldots \left[\sum_{n_{R-1,0}=0}^{J_{R-1}} h(R\text{-}1;J_{R-1}\text{-}n_{R-1,0}) \left[\sum_{n_{R,0}=0}^{J_R} h(R;J_R\text{-}n_{R,0})\, n_0! \right] \right] \ldots \right].$$

Defining three temporary vectors as follows:

$T = (t_1, \ldots, t_R)$ is the accumulated sum of job class distribution, where

$$t_r = \sum_{i=1}^{r} n_{i,0} \quad , \text{ note: } t_R = n_0 \quad ,$$

$W = (w_1, \ldots, w_R)$ is the accumulated factorial values, where

$$w_r = t_r! \quad , \text{ note: } w_R = n_0! \quad ,$$

$V = (v_1, \ldots, v_R)$ is the accumulated summation value of the product terms, where

$$v_i = \sum_{n_{i,0}=0}^{J_i} h(i;J_i-n_{i,0}) \left[\ldots \left[\sum_{n_{R-1,0}=0}^{J_{R-1}} h(R-1;J_{R-1}-n_{R-1,0}) \left[\sum_{n_{R,0}=0}^{J_R} h(R;J_R-n_{R,0})\, n_0 ! \right] \right] \ldots \right].$$

Thus $v_1 = G(J)$, and v_R is the innermost product term. The SOP algorithm proceeds as follows:

BEGIN: SOP algorithm

STEP 1: initialize 1-st vector elt.

$i = 1$

$n_{i,0} = 0$

$t_i = 0$

$w_i = 0$

$v_i = 0$

STEP 2: compute remaining vector elts.

STEP BY 1 j=i+1 TO R

BEGIN

$n_{j,0} = 0$

$t_j = t_{j-1}$

$w_j = w_{j-1}$

$v_j = 0$

END

STEP 3: compute inner-most product term

STEP BY 1 $n_{R,0} = 0$ TO J_R

BEGIN

$v_R = v_R + w_R * h(R;J_R-n_{R,0})$

$t_R = t_R + 1$

$w_R = w_R * t_R$

END

$i = R$

STEP 4: expand outward accumulating product term sums

$i = i-1$

IF i<1 GO TO STEP 6

$v_i = v_i + v_{i+1} * h(i;J_i-n_{i,0})$

$n_{i,0} = n_{i,0} + 1$

STEP 5: update factorial if "sum-limit" not reached

IF $n_{i,0} > J_i$ GO TO STEP 4

$t_i = t_i + 1$

$w_i = w_i * t_i$

GO TO STEP 2

STEP 6: terminate algorithm

STOP

END: SOP algorithm

The SOP algorithm requires storage for three temporary vectors (T, V, W), each containing R elements, and for the R vectors of $h(i,J_i)$, each containing J_i+1 elements. Therefore, the total storage required for this algorithm is

$$3R + \sum_{i=1}^{R} (J_i+1) = 4R + \sum_{i=1}^{R} J_i = 4R + K \quad .$$

To determine the number of operations needed to evaluate the SOP equation form, note that one addition and one multiplication are required for each combination of the first (outer-most) R-1 product terms. For each of these combinations the entire inner-most (R-th) product term and the factorial must be evaluated, requiring two multiplications and one addition at each step. This results in an operation count on the order of

$$\left[\prod_{i=1}^{R-1} (J_i+1) \right] \left[2 + 3(J_R+1) \right] = \left[5 + 3J_R \right] \left[\prod_{i=1}^{R-1} (J_i+1) \right] \quad .$$

The FAC expansion requires an additional algorithm to sum over every state in a constrained sub-space, which yields all combinations of different job classes keeping the total number of jobs constant. The sub-space is defined by all solutions to

$$\sum_{i=1}^{R} n_{i,0} = k \quad ,$$

with the constraint of

$$n_{i,0} \leq J_i \quad , \qquad \forall\, i \quad .$$

To sum over the constrained sub-space start with class 1 jobs. Next from a total of k jobs determine the maximum and minimum number of jobs that can be allocated to classes 2 thru R in conjunction with the constriants, $0 \leq n_i \leq J_i$. Then determine the allowable range of class 1 jobs based on the maximum and minimum values just computed. Stepping through the range of class 1 jobs, determine the maximum and minimum number of jobs that can be allocated to classes 3

through R from the remaining $k-n_{1,0}$ jobs. Then compute the allowable range of class 2 jobs. Continuing this procedure for each job class results in the following state-space generation process

$$\sum_{n_{1,0}=MAX[0,q_1]}^{MIN[J_1,m_1]} \cdots \sum_{n_{R,0}=MAX[0,q_R]}^{MIN[J_R,m_R]} ,$$

where m_i represents the number of jobs to be distributed over queues i thru R, and is expressed as

$$m_i = \begin{cases} m_{i-1}-n_{i-1} , & R \geq i > 1 \\ k , & i = 1 , \end{cases}$$

and q_i represents the minimum number of jobs that must be placed in the i-th queue (which may be negative), and is expressed as

$$q_i = \begin{cases} m_i - t_{i+1} , & i < R \\ m_i , & i = R . \end{cases}$$

Utilizing the above, the complete FAC expansion can be formulated as

$$\sum_{k=0}^{K} k! \left[\sum_{n_{1,0}=MAX[0,q_1]}^{MIN[J_1,m_1]} h(1;J_1-n_{1,0}) \cdots \sum_{n_{R-1,0}=MAX[0,q_{R-1}]}^{MIN[J_{R-1},m_{R-1}]} h(R-1;J_{R-1}-n_{R-1,0})\, h(R;J_R-n_{R-1,0}+m_{R-1}) \right] .$$

Defining five temporary vectors as follows:

$M=(m_1, \ldots, m_R)$ the maximum set of jobs to be distributed over queues i to R, where

$$m_i = \begin{cases} m_{i-1}-n_{i-1} , & R \geq i > 1 \\ k , & i = 1 , \end{cases}$$

$Q=(q_1, \ldots, q_R)$ is the minimum number of jobs that the i-th queue can accept, where

$$q_i = \begin{cases} m_i - t_{i+1} & , \quad i < R \\ m_i & , \quad i = R \end{cases} ,$$

$T=(t_1, \ldots, t_R)$ is the maximum number of jobs that may be allocated to queues r to R, where

$$t_r = \sum_{i=R}^{r} J_i \quad , \quad \text{note: } t_1 = K ,$$

$V=(v_1, \ldots, v_R)$ is the accumulated summation of the product terms, where

$$v_i = \begin{cases} \sum_{k=0}^{K} k! \left[\sum_{\substack{R \\ \sum_{i=1} n_{i,0}=k}} \prod_{i=1}^{R} h(i;J_i-n_{i,0}) \right] & , \quad i=1 \\ \sum_{n_{i-1,0}=MAX[0,q_{i-1}]}^{MIN[J_{i-1},m_{i-1}]} h(i-1;J_{i-1}-n_{i-1,0}) \quad \ldots \quad \sum_{n_{R,0}=MAX[0,q_R]}^{MIN[J_R,m_R]} h(R;J_R-n_{R,0}) & , \quad i>1, \end{cases}$$

$UP=(up_1, \ldots, up_R)$ is the maximum number of jobs that can be allocated to the i-th queue, where

$$up_i = MIN[J_i, m_i] \quad .$$

The FAC algorithm proceeds as follows:

```
BEGIN: FAC algorithm
       STEP 1: initialize
               fac=1
               t_R=J_R
               v_R=0
               k=-1
               STEP BY -1  i=R-1  TO  1
               BEGIN
                       v_i=0
                       t_i=t_{i+1}+J_i
               END
```

STEP 2: compute 1-st elt of temporary distribution vectors

$k = k+1$

IF $k > t_1$ GO TO STEP 8

$m_1 = k$

$q_1 = m_1 - t_2$

$n_{1,0} = MAX[0, q_1]$

$up_1 = MIN[J_1, m_1]$

$i = 2$

STEP 3: compute remaining vector elements

STEP BY 1 $r = i$ TO $R-1$

BEGIN

$m_r = m_{r-1} - n_{r-1}$

$q_r = m_r - t_{r+1}$

$n_{r,0} = MAX[0, q_r]$

$up_r = MIN[J_r, m_r]$

END

STEP 4: compute inner-most product term

STEP BY 1 $n_{R-1,0} = n_{R-1,0}$ TO up_{R-1}

BEGIN

$n_{R,0} = m_{R-1} - n_{R-1,0}$

$v_R = v_R + h(R-1; J_{R-1} - n_{R-1,0}) * h(R; J_R - n_{R,0})$

END

$i = R$

STEP 5: expand outward accumulating product term sums

$i = i-1$

IF $i < 2$ GO TO STEP 7

$v_i = v_i + v_{i+1} * h(i-1; J_{i-1} - n_{i-1,0})$

$v_{i+1} = 0$

$n_{i-1,0} = n_{i-1,0} + 1$

STEP 6: test if "sum-limit" reached

IF $n_{i-1,0} < up_{i-1}$ GO TO STEP 3

GO TO STEP 5

STEP 7: accumulate outer-sum term & update factorial

$v_1 = v_1 + v_2 * fac$

$fac = (k+1) * fac$

GO TO STEP 2

STEP 8: terminate algorithm

STOP

END: FAC algorithm

Solution Method	Storage Requirements	Computation Requirements
SOP	$4R + K$	$[5 + 3J_R] \left[\prod_{i=1}^{R-1} (J_i + 1) \right]$
FAC	$6R + K$	$2 \left[\prod_{i=1}^{R} (J_i + 1) \right] + 3K$
SCS Iterative	$\prod_{i=1}^{R} (J_i + 1)$	$4R \left[\prod_{i=1}^{R} (J_i + 1) \right]$
General Iterative	$\prod_{i=1}^{R} (J_i + 1)$	$2(R + L - 1) \left[\prod_{i=1}^{R} (J_i + 1) \right]$

Figure III-4. Storage and computation complexity.

Solution Method	Storage Requirements	Computation Requirements
SOP		
example 1	18	120
example 2	54	155,520
FAC		
example 1	22	102
example 2	66	93,402
SCS Iterative		
example 1	36	288
example 2	46,656	1,119,744
General Iterative		
example 1	36	576
example 2	46,656	2,239,488

Example 1:

R = 2
S = (1,3,3)
K = 10
J = (5,5)

Example 2:

R = 6
S = (1,3,3,3,3,3,3)
K = 30
J = (5,5,5,5,5,5)

Figure III-5. Example of storage and computation complexity.

The FAC algorithm requires storage for five temporary vectors (M, Q, T, V, UP), each containing R elements, and for the R $h(i;J_i)$ vectors, each containing J_i+1 elements. Therefore, the total storage required for this algorithm is

$$5R + \sum_{i=1}^{R} (J_i+1) = 6R + \sum_{i=1}^{R} J_i = 6R + K \quad .$$

The total number of states in the state space is

$$\prod_{i=1}^{R} (J_i+1) \quad .$$

The number of steps carried out for the inner product terms is equal to the total number of states. Each step (a value of k) requires one addition and one multiplication. The outer term for each step requires two multiplications and one addition. This results in an operation count on the order of

$$\left[2 \prod_{i=1}^{R} (J_i+1)\right] + \left[3 \sum_{i=1}^{R} J_i\right] = \left[2 \prod_{i=1}^{R} (J_i+1)\right] + 3K$$

Figure III-4 provides a summary of the storage and computational requirements of the SOP and FAC algorithms, as well as those for the general iterative procedure for multi-class queueing networks and for that procedure adapted to the SCS model. The general iterative procedure is adapted to the SCS model (SCS iterative) by representing each ICS as a single equivalent device [CHANDY 75B, GIAMMO 76], therefore, the equivalent number of devices L-1 now becomes R. Note that the storage requirements for the SOP and FAC algorithms increase linearly with the number and distribution of jobs, K and J, and ICSs, R; while the storage requirements of previous algorithms increase exponentially. In figure III-5 these requirements are evaluated for two examples; the first is a small network of 2 computer systems (or job classes) with a total of 7 devices and with 10 jobs equally allocated between the 2 systems; the second is a moderate network comprising 6 computer systems (or job classes) with a total of 19 devices and with 30 jobs equally allocated among the 6 systems.

It can be seen that the SOP and FAC algorithms require very little storage compared to both iterative algorithms, while also requiring fewer computations. In addition, the mean queue length for all devices in the SCS model can be computed using either of the algorithms; whereas, in the general case no effective procedure yet exists. Although it should be noted that by using the iterative procedures relatively little additional computation is needed to obtain the device busy probability and the mean queue length for all but the SPR. Computing these measures using the SOP or FAC algorithms entails a larger amount of computation, but also includes evaluation of performance measures for the SPR. Comparing the SOP and FAC algorithms one can see from figures III-4 and III-5 that the SOP algorithm uses less storage while the FAC algorithm requires fewer computational steps (i.e. less time).

To complete this discussion we shall present our expressions for the performance measures of the SCS model, equations (36) through (45), restructured into forms readily evaluated by the FAC or SOP algorithms.

Normalization constant

$$(46)\quad G(J) = \sum_{n_{1,0}=0}^{J_1} h(1;J_1-n_{1,0}) \left[\ldots \left[\sum_{n_{R-1,0}=0}^{J_{R-1}} h(R-1;J_{R-1}-n_{R-1,0}) \left[\sum_{n_{R,0}=0}^{J_R} h(R;J_R-n_{R,0})\, n_0 ! \right] \right] \ldots \right] , \text{ or}$$

$$(47)\quad = \sum_{n_0=0}^{K} n_0 ! \left[\sum_{n_{i,0}=MAX[0,q_i]}^{MIN[J_i,m_i]} h(i;J_i-n_{i,0})\, F_i(n_0-n_{i,0}) \right] , \text{ for any } i ,$$

where,

$$F_i(n_0-n_{i,0}) = \sum_{\substack{R \\ \Sigma n_{i,0}=n_0-n_{r,0} \\ r=1\ ,\neq i}} \prod_{r=1\ ,\neq i}^{R} h(r;J_r-n_{r,0}) \quad , \text{ and}$$

$$q_i = n_0 - \sum_{r=1\ ,\neq i}^{R} J_r \quad .$$

Device busy probability

$$(48)\quad A_{i,j} = \frac{x_{i,j}}{G(J)} \sum_{n_{1,0}=0}^{J_1} h(1;J_1-n_{1,0}) \left[\ldots \left[\sum_{n_{i,0}=0}^{J_i-1} h(i-1;J_i-1-n_{i,0}) \ldots \left[\sum_{n_{R-1,0}=0}^{J_{R-1}} h(R-1;J_{R-1}-n_{R-1,0}) \right.\right.\right.$$

$$\left.\left.\left. \left[\sum_{n_{R,0}=0}^{J_R} h(R;J_R-n_{R,0}) n_0 ! \right] \right] \ldots \right] \ldots \right] \quad , \text{ or}$$

$$(49)\quad A_{i,j} = \frac{x_{i,j}}{G(J)} \sum_{n_0=0}^{K-1} n_0 ! \left[\sum_{n_{i,0}=MAX[0,q_i]}^{MIN[J_i-1,m_i]} h(i;J_i-1-n_{i,0})\, F_i(n_0-n_{i,0}) \right] ,$$

$$(50)\quad A_0 = P[n_0 \geq 1] = 1 - \prod_{i=1}^{R} \frac{g_i(J_i)}{G(J)} \quad .$$

Mean queue length

$$(51)\quad Q_{i,j} = \frac{1}{G(J)} \sum_{k=1}^{J_i} x_{i,j}^{k} \left[\sum_{n_{1,0}=0}^{J_1} h(1;J_1-n_{1,0}) \left[\ldots \left[\sum_{n_{i-1,0}=0}^{J_{i-1}-k} h(i-1;J_{i-1}-n_{i-1,0}) \ldots \right.\right.\right.$$

$$\left.\left.\left. \left[\sum_{n_{R-1,0}=0}^{J_{R-1}} h(R-1;J_{R-1}-n_{R-1,0}) \left[\sum_{n_{R,0}=0}^{J_R} h(R;J_R-n_{R,0}) n_0 ! \right] \right] \ldots \right] \ldots \right] , \text{ or}$$

$$(52)\quad Q_{i,j} = \frac{1}{G(J)} \sum_{k=1}^{J_i} x_{i,j}^{k} \left[\sum_{n_0=0}^{K-k} n_0 ! \left\{ \sum_{n_{i,0}=MAX[0,q_i]}^{MIN[J_i-k,n_0]} h(i;J_i-k-n_{i,0})\, F_i(n_0-n_{i,0}) \right\} \right] , \text{ and}$$

$$(53)\quad Q_0 = \frac{1}{G(J)} \sum_{n_{1,0}=0}^{J_1} h(1;J_1-n_{1,0}) \left[\ldots \left[\sum_{n_{R-1,0}=0}^{J_{R-1}} h(R-1;J_{R-1}-n_{R-1,0}) \left[\sum_{n_{R,0}=0}^{J_R} h(R;J_R-n_{R,0})\, n_0 !\, n_0 \right] \right] \ldots \right] .$$

$$(54) \qquad Q_0 = \frac{1}{G(J)} \sum_{n_0=1}^{K} n_0! \left[n_0 \sum_{n_{i,0}=MAX[0,q_i]}^{MIN[J_i,m_i]} h(i;J_i-n_{i,0})\, F(n_0-n_{i,0}) \right], \text{ and}$$

$$(55) \qquad Q_{i,0} = \frac{1}{G(J)} \sum_{n_0=1}^{K} n_0! \left[\sum_{n_{i,0}=MAX[0,q_i]}^{MIN[J_i,m_i]} n_{i,0}\, h(i;J_i-n_{i,0})\, F(n_0-n_{i,0}) \right]$$

The device throughput

$$(56) \qquad T_{i,j} = u_{i,j} A_{i,j} = \frac{G(J-d_i)}{G(J)} e_{i,j} \quad , \quad \text{for } i,j \neq 0 \quad , \text{ and}$$

$$(57) \qquad T_0 = u_0 A_0 = u_0 \left\{ 1 - \frac{\prod_{i=1}^{R} g_i(J_i)}{G(J)} \right\} .$$

These expressions can be computed simultaneously in groups, equations (46) and (53), or (47) and (54) comprise one group, and (49), (52) and (55) another. Once the values for these performance measures are computed they can then be applied to directly evaluate the remaining equations, (50), (56), and (57). The FAC and SOP algorithms can be modified to compute each group at the same time; this is especially useful for the later group which can share intermediate values (e.g. $F_i(n_0-n_{i,0})$) and, therefore, eliminate duplicate computations. A Fortran implementation of these algorithms was developed and is used later in this dissertation to compute values for these performance measures.

IV. APPROXIMATE SCS MODEL

A. The Approximation

Efficient algorithms for queueing networks have been previously developed [BUZEN 73, MUNTZ 74, SHUM 76], and a new algorithm that is very efficient in its memory space requirements has been presented here in chapter III. Still, it can be seen from Figure III-4, that the computation time is a significant burden; it is of exponential complexity and, therefore, computationally intractable. In addition, the complex form of the equations conveys little useful intuitive information or discernable insight.

Some previous efforts have concentrated on developing approximate solutions for various models. Reducing the computation and memory-space complexity, or generalizing the modeling assumptions are the primary motivations. These generalizations include more general service time distributions, accounting for passive resources, simultaneous acquisition of multiple resources, resource blocking, priority and other scheduling policies, state dependent routing, and others [CHANDY 78].

Kobayashi [KOBAYA 74A] has utilized the diffusion approximation to model queueing networks with general service time distributions, assuming a Poission arrival process and a FCFS scheduling policy. This approach has the potential to investigate the network transient state behavior [KOBAYA 74B]. The diffusion approximation is primarily applicable to open networks and currently has limited utility for a closed network. Chandy [CHANDY 75A] has introduced an aggregation technique similar to Norton's theorem in electrical circuits. This technique allows one to represent a number of queues as a single equivalent queue. If the queue satisfies local balance [CHANDY 72B], then the technique yields exact solutions; if not, a similar technique with an additional flow approximation procedure yields approximate solutions [CHANDY 75B]. These techniques are of computation time and memory-space complexity equivalent to those of the convolution algorithm of basic queueing network theory [BUZEN 73, MUNTZ 74, SHUM 76].

Other approximation efforts have studied the effect of substituting one queueing type for another. Buzen [BUZEN 74] investigated using a mathematically less complex M/G/1 service center to approximate an M/G/1/K service center. In a later effort Buzen [BUZEN 77] approximated an M/G/1/K service center by using an M/M/1/K service center. Buzen's efforts were directed towards a single service center; whereas, Shum [SHUM 76] investigated the substitution of M/G/1 product terms for M/M/1 product terms in an effort to approximate general service time distributions in a multi-class queueing network.

Avi-Itzhak [AVI-IT 73] used a conservation of flow argument to establish an expression for the mean burst cycle time in a central server model. This expression requires the mean number of busy servers (busy probability) at a central server, which must be obtained by solving the queueing network equations and summing over the entire state space. He then used this parameter along with an assumed geometric cycle distribution as an approximation to the queue dependent mean service rate in a single server queue. Solving the basic state balance equations, assuming the arrival process is Poisson, results in expressions for waiting and delay times for the system.

A major obstacle in using the basic queueing equations as approximations to queueing networks is the difficulty of relating the corresponding input parameters of the basic queueing equations to those of queueing networks. Queueing networks require the mean service rate of each device, u_i, transition probabilities between devices, p_{ij}, and the number of jobs constantly circulating in the network, K. The basic single server queueing equations require the same first parameter, but utilize arrival rate as the other.

We shall utilize a similar conservation of flow argument as Avi-Itzhak to establish a relationship between arrival rate and the number of jobs in the network. From this we shall utilize independent single server queues to approximate the behavior of the SCS queueing network model.

Assuming that each device of the SCS is an M/M/1 single server queue, it can be shown [BURKE 56, FINCH 59, BURKE 72, MUNTZ 73, KLIENR 76] that the arrival process is equivalent to the departure process. The arrival process in an M/M/1 queue is Poisson with parameter a, therefore, the mean flow rate in is equal to the mean flow rate out:

$$\text{rate}_{in} = \text{rate}_{out} = a \quad .$$

For the CPU in each ICS the flow out is decomposed into separate Poisson flows, which proceed to the various PPUs and the SPR. The decomposition of a Poisson flow in this manner is linear [COFFMA 73, pg 149-150]. For the SCS this results in

$$(1) \qquad a_{CPU_i} = \sum_{j=2}^{s_i} a_{i,j} + a_{SPR_i} \quad , \text{ and}$$

$$(2) \qquad a_{i,j} = p_{i,j}\, a_{CPU_i}$$

$$a_{SPR_i} = p_{SPR_i}\, a_{CPU_i} = p_{i,0}\, a_{CPU_i} \quad , \text{ and} \quad a_{SPR} = \sum_{i=1}^{R} a_{SPR_i} \quad ,$$

where the following subscript notation is adopted for clarity

$$SPR_i = i,0$$
$$CPU_i = i,1$$
$$PPU_{i,j} = i,j \qquad j>1 \quad .$$

Having established a flow relationship between devices, a relationship between the queueing network parameter K and the independent single sever queueing parameter $a_{i,j}$ is necessary. For an M/M/1 queue the mean queue length (including a job in service), given its mean arrival (a) and service (u) rates, is [KLIENR 75, KLIENR 76, COFFMA 73]

$$(3) \qquad Q = \frac{1}{1/\rho - 1} \quad ,$$

where $\rho = a/u$. By assuming each device is an independent M/M/1 single server queue we may use (2) to establish the following relation

$$(4) \qquad K = \sum_{i=1}^{R} \left\{ \frac{1}{1/\rho_{SPR} - 1} + \frac{1}{1/\rho_{CPU_i} - 1} + \sum_{j=2}^{s_i} \frac{1}{1/\rho_{PPU_{i,j}} - 1} \right\} \quad ,$$

$$(5) \qquad J_i = \frac{\alpha_i}{1/\rho_{SPR} - 1} + \frac{1}{1/\rho_{CPU_i} - 1} + \sum_{j=2}^{s_i} \frac{1}{1/\rho_{PPU_{i,j}} - 1} \quad ,$$

where

$$\rho_{SPR} = \frac{\sum_{i=1}^{R} p_{SPR_i} a_{CPU_i}}{u_{SPR}} \quad ,$$

$$\rho_{PPU_{i,j}} = \frac{p_{i,j} a_{CPU_i}}{u_{i,j}} \quad ,$$

$$\rho_{CPU_i} = \frac{a_{CPU_i}}{u_{CPU_i}} \quad , \text{ and}$$

$$\alpha_i = \frac{p_{SPR_i} a_{CPU_i}}{\sum_{r=1}^{R} p_{SPR_r} a_{CPU_r}} \quad .$$

Assuming futher that each ICS is identical (i. e. a balanced system), this then yields

$$(6) \qquad J_i = \frac{1/R}{1/\rho_{SPR} - 1} + \frac{1}{1/\rho_{CPU_i} - 1} + \sum_{j=2}^{s_i} \frac{1}{1/\rho_{PPU_{i,j}} - 1} \quad .$$

B. Computational Algorithm and Performance Measures

Given the queueing network parameters (J_i, $p_{i,j}$, and $u_{i,j}$) one may approximate the flow rate, a_{CPU_i}, by solving (6). Although (6) is an equation with a single unknown and not computationally complex, it does not lend itself to a closed form analytic solution. We shall present an algorithm, utilizing a bounded binary search technique, to efficiently solve (6) for a_{CPU_i}. Rewriting (6) results in

$$(7) \qquad J_i = \frac{1/R}{1/(a_{CPU_i} x_{SPR}) - 1} + \frac{1}{1/(a_{CPU_i} x_{CPU_i}) - 1} + \sum_{j=2}^{s_i} \frac{1}{1/(a_{CPU_i} x_{PPU_{i,j}}) - 1},$$

where

$$x_{SPR} = \frac{R\, p_{SPR_i}}{u_{SPR}},$$

$$x_{PPU_{i,j}} = \frac{p_{i,j}}{u_{i,j}}, \text{ and}$$

$$x_{CPU_i} = \frac{1}{u_{CPU_i}}.$$

Without loss of generality assign $u_{CPU_i} = 1$. This produces a normalizing effect, allowing all other service rates to be stated relative to this standard unit of service. A lower bound for flow rate is zero, and from inspection of (7) an upper bound is MAX[x_{SPR}, x_{CPU_i}, $x_{PPU_{i,2}}$, ... , $x_{PPU_{i,s_i}}$]. Using these flow rate bounds, a binary search technique may be used to approach the flow rate that will satisfy (7) to within some arbitrary error δ. This algorithm,

the BIN algorithm, is stated more formally below. Because systems are balanced for i=1, ... , R (i.e. identical) we perform the following on the arbitrarily chosen system i=1.

BEGIN BIN

STEP 1: compute initial parameters

$x_{SPR} = (R\, p_{SPR})/u_{SPR}$

$x_{CPU} = 1$

STEP BY 1 j=2 TO s_i

$x_{PPU_{1,j}} = p_{1,j}/u_{1,j}$

END

STEP 2: set initial search bounds

low = 0

high = $MAX[x_{SPR}, x_{CPU}, x_{PPU_{1,2}}, \ldots, x_{PPU_{1,s_i}}]$

STEP 3: evaluate at midpoint of bounds

mid = (low+high)/2

val = $(1/R)/(1/(\text{mid } x_{SPR})-1) + 1/(1/(\text{mid } x_{CPU})-1)$

STEP BY 1 j=2 TO s_i

val = val + $1/(1/(\text{mid } x_{PPU_{1,j}})-1)$

END

STEP 4: convergence test and adjust bounds

IF($|val-J_1| \leq \delta$) GO TO STEP 5

IF($val < J_1$) low = mid

If ($val > J_1$) high = mid

GO TO STEP 3

STEP 5: terminate with flow rate = mid

STOP

END BIN

The BIN algorithm requires storage for the vector $\dot{X}$, containing s_i+1 elements, the convergence error, and the instantaneous solution along with its corresponding search region description (bounds and midpoint). Therefore, the total storage required for this algorithm is

$$(s_i+1) + 1 + (1+3) = s_i+6$$

The number of operations necessary to evaluate (7) using the BIN algorithm depends on the number of iterations required, which is a function of the convergence error. For $2^{-(n+1)} < \delta \leq 2^{-n}$, the maximum number of iterations is n. Each iteration requires $4+5\, s_i$ operations, thus requiring an operation count $O[\, n(4+5\, s_i)\,]$. Comparing these complexities with those in figure III-4 of chapter III, one can see the significant advantage of this approximation over the exact model.

The BIN algorithm applied to (7) allows one to determine the job flow rate for a given set of queueing network parameters. Once this is done then the performance measures for each device may be easily computed using the following [COFFMA 73, KLIENR 75] :

Device busy probability

$$\Pr[n_{i,j} > 0] = \rho_{i,j} \quad , \tag{8}$$

$$\Pr[n_0 > 0] = \rho_{SPR} \quad ,$$

Device mean queue length

$$Q_0 = \frac{1}{1/\rho_{SPR} - 1} \quad , \tag{9}$$

$$Q_{i,j} = \frac{1}{1/\rho_{PPU_{i,j}} - 1} \quad , \quad i,j > 0 \quad ,$$

Device throughput

$$T_0 = u_{SPR}\ \rho_{SPR} \quad , \tag{10}$$

$$T_{i,j} = u_{i,j}\ \rho_{i,j} \quad , \quad i,j > 0 \quad .$$

Note, throughput may more properly be referred to as the effective service rate or departure rate of the device, which for an M/M/1 queue equals a, the arrival rate.

The mean cycle time of a job is the mean time (wait or delay) between successive requests to the CPU by the same job. This is the weighted sum of the mean time it takes a job to be serviced at each device. Using Little's formula (W=Q/a) this may be computed as

$$W_i = e_{SPR_i} W_{SPR} + W_{CPU_i} + \sum_{j=2}^{s_i} e_{PPU_{i,j}} W_{PPU_{i,j}} \tag{11}$$

$$= \frac{e_{SPR_i} Q_{SPR}}{T_{SPR}} + \frac{Q_{CPU_i}}{a_{CPU_i}} + \sum_{j=2}^{s_i} \frac{e_{PPU_{i,j}} Q_{PPU_{i,j}}}{T_{PPU_{i,j}}}$$

$$= \frac{e_{SPR_i} Q_{SPR}}{R\, p_{SPR_i}\, a_{CPU_i}} + \frac{Q_{CPU_i}}{a_{CPU_i}} + \sum_{j=2}^{s_i} \frac{e_{PPU_{i,j}} Q_{PPU_{i,j}}}{a_{CPU_i}\, p_{PPU_{i,j}}}$$

$$= \frac{1}{a_{CPU_i}} \left\{ Q_{SPR}/R + Q_{CPU_i} + \sum_{j=2}^{s_i} Q_{PPU_{i,j}} \right\}$$

$$= J_i / a_{CPU_i} \quad ,$$

where the relative visit frequency is (note: $u_{CPU_i} = 1$)

$$e_{i,j} = \begin{cases} p_{SPR_i} u_{CPU_i} = p_{SPR_i} \\ \\ p_{PPU_{i,j}} u_{CPU_i} = p_{PPU_{i,j}} \end{cases} .$$

An approximate analysis technique has been presented for a balanced SCS model which is much less complex to evaluate compared to existing efficient queueing network technique. The question remains as to the error this approximation introduces, and a justification for the choice of an M/M/1 queue.

An M/M/1/K queue is an M/M/1 with a finite queue length, and intuitively would seem to better approximate the operations of the individual devices of a closed network. We have investigated the use of this well known queue, and typical results for device throughput and mean queue length are presented in figures IV-1 and IV-2. As can be seen from these figures the M/M/1/K queue did not produce significantly better results than the M/M/1 queue for the examples considered. Generally, the most important aspect of using these models concerns when and how these curves react to variations in parameters. Little , if any, significance is associated with the absolute values of these curves, except on a relative basis. This implies that the primary importance of any approximation is in "tracking" the actual curve rather than replicating it. As can be seen from the figures both the M/M/1 and the M/M/1/K approximations track the exact (SCS) queueing network results.

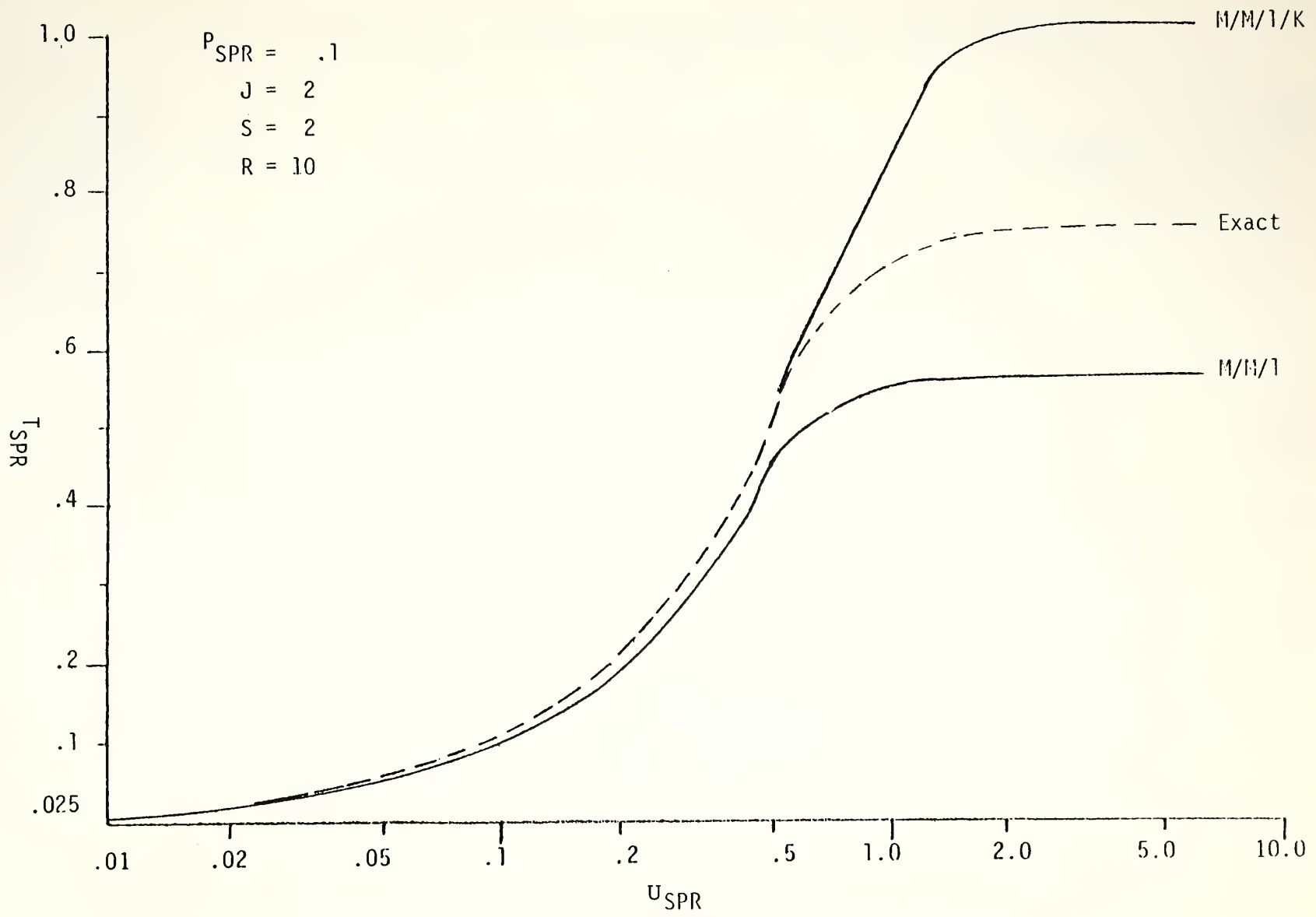

Figure IV-1. M/M/1/K and M/M/1 throughput comparison.

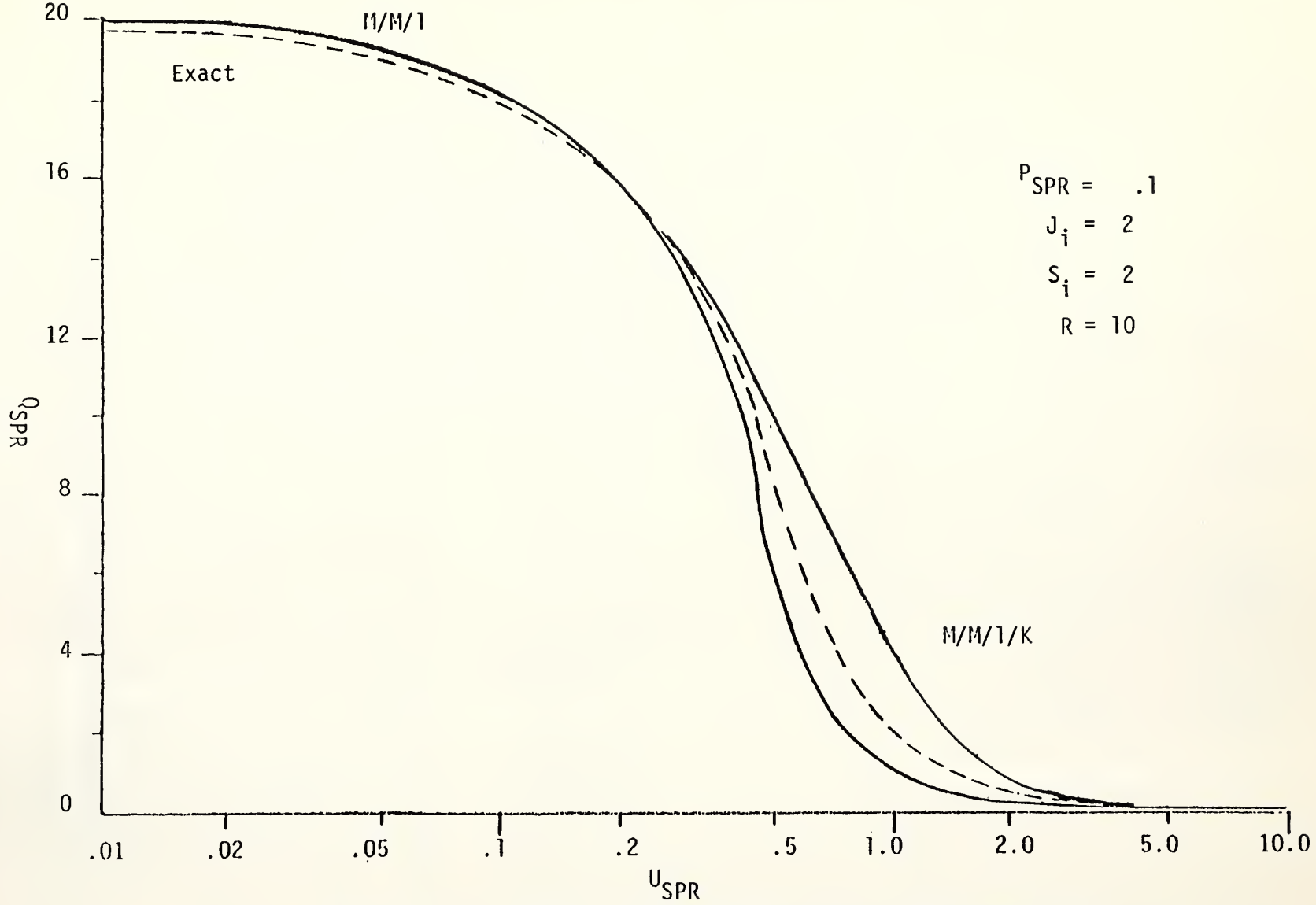

Figure IV-2. M/M/1/K and M/M/1 mean queue length comparison.

The M/M/1/K approximation does not yield a significantly better fit to the exact curve than does the M/M/1 approximation; therefore, the M/M/1 approximation was selected for the following reasons. First, as mentioned earlier, the major concern is tracking the exact curve and not in duplicating it. Since both curves track well, choosing the best fit was not necessary. Second, the computational complexity of the M/M/1/K approximation is greater than that of the M/M/1. The M/M/1/K expression for the mean queue length [ALLEN 78] corresponding to (3) is

$$Q = \frac{\rho\,[\,1-(K+1)\rho^{K}+K\rho^{(K+1)}\,]}{(1-\rho)\,(1-\rho^{(K+1)})}$$

$$= \frac{1}{1/\rho-1} - \frac{K+1}{(1/\rho)^{(K+1)}-1} \quad .$$

In a practical situation these expressions are evaluated by a computational device (computer or calculator), which introduces errors due to the use of approximation algorithms for exponentiation and the lack of precision (bits) when the queue approaches saturation. Also the lower computational complexity of evaluating (3) allows one to gain insight into the systems' operation directly from the form of the equations.

Buzen [BUZEN 74] has compared the M/G/1 queue as an approximation to the M/G/1/K queue. He has determined that except for heavy traffic ($\rho \simeq 1$) and small queue capacity ($K \simeq 1$) that the relative error is small. Using arguments similar to the ones presented here, Buzen recommends the use of the M/G/1 as a reasonable approximation to the M/G/1/K queue.

C. Error Analysis of Approximation

The sensitivity and magnitude of the error introduced by our approximate SCS model compared to the exact SCS model is investigated. Since the form of the exact model is so mathematically complex, a direct analytical comparison is not feasible. The alternative is to numerically evaluate the two models for corresponding parametric values and compare the results.

The problem in attempting this is that the combination of all possible parametric values is infinitely large. Therefore, a reasonable and representative subset of values will be selected. Using Fortran programs developed to implement our algorithms, and this set of chosen parametric values we will compare the performance measures presented in the previous section, specfically, the throughput and mean queue length of the CPU and SPR. Note, that the throughput measure may more correctly be referred to as effective departure rate. Since the device busy probability performance measure is directly related to the throughput by a constant, comparing either one to the corresponding exact value would yield identical results. A balanced system(identical ICSs) is assumed for simplicity.

Due to the assumption of a balanced system we may drop the added burden of carrying extra subscripts to distinguish between individual ICSs, as the notation below indicates. This notation simplification results in previously defined vector elements (i.e. J_i and s_i) now being denoted by their vector notation (i.e. J and s). For both the exact and approximate models the following are the pertinent parameters and their complete allowable ranges (see Appendix C for the simplified notation):

$$0 < R < \infty$$

$$0 < K = RJ = RJ_i < \infty \quad , \quad i = 1, \ldots, R$$

$$0 < s = s_i < \infty \quad , \quad i = 1, \ldots, R$$

$$0 < u_j = u_{ij} < \infty \quad , \quad i = 1, \ldots, R \text{ and } j = 0, \ldots, s_i$$

$$0 < p_j = p_{ij} < 1 \quad , \quad i = 1, \ldots, R \text{ and } j = 0, \ldots, s_i \ .$$

All of the parameters, with the exception of the transition probabilities, each have an infinitely large range, as can be seen above. The selection of a small, finite subset of each of them to form a manageable sample space shall now be discussed.

Both the number of ICSs, R, and the number of jobs per ICS, J, have a significant impact on the computational complexity of the exact SCS model. From chapter III, the computation tine complexity is $O[(J+1)^R]$. Because of our interest in modular expansion, R is felt to be slightly more important. Therefore, our selection put more emphasis on R than J. From initial testing and experimentation, we found that the processing time for the configuration of $J = 2$ and $R = 8$ on a DEC 10 computer was approximately 1.4 minutes. Based on this we selected 8 as the maximum

value of R and subsequently 6 as the maximum value of J. Further selection of additional elements to construct representative sets large enough to provide insight into developing trends resulted in J = {2, 4, 6} and R = {1, 2, 5, 8}.

The number of devices within an ICS, s, does not present a significant computational problem. In the exact model all of the devices within each ICS are "collapsed" into a single equivalent device. A reasonable upper limit might be 11 devices per ICS, comprising a CPU and 10 PPUs. For a large number of configurations this would provide for a sufficient number of PPUs. For these configurations and, also for larger ones, a representative set large enough to provide insight into any developing trends is {2, 6, 11}.

The two remaining parameters, processing rates and transition probabilities, differ from the others in that a single value is not a sufficient specification. A group of values for each parameter is required, one for each device within an ICS as well as the SPR. The value of either of these parameters does not itself impact the computational effort required, although each group of values requires a separate computation, as does a change of any parameter. A finite subset of values for each of these two parameters will be first selected, and a procedure to be used to assign these values to the devices will be discussed.

Since the processing rate of the CPU has been fixed at unity, all other processing rates are relative to the CPU. A relative range spanning 3 orders of magnitude from .01 to 10.0 provides a representative range. The processing rates are important parameters of the model. Contrasting their importance is the need to minimize the sample space. As a compromise, we selected a relatively large number of values, 10. We have selected the set {.01, .02, .05, .1, .2, .5, 1.0, 2.0, 5.0, 10.0}.

Each device transition probability by definition is bounded between 0 and 1, and the sum of all transition probabilities from each CPU is constrained to equal unity. A representative selection must span the bounded range, therefore, {.1, .25, .5, .75, .9} has been selected.

The assignment procedure we will follow is to select a transition probability value from the subset and assign it to the SPR, p_{SPR}. The remaining probability, $1 - p_{SPR}$, will be randomly

distributed among each of the remaining s devices. Also at the same time a device transition probability is assigned, the relative processing rate will also be assigned by random selection from the subset of relative processing rates.

The details of this procedure are discussed here and the algorithm is presented below. For each device divide the remaining probability into two groups. The first group is a reserve, of 25%, to assure that any remaining devices are allocated some probability. The other group, representing the bulk of the probability, is the selection range for the current device. Generate a random number in the continuous, open interval (0,1), from a uniform probability distribution. Multiply this fraction by the upper value of the probability selection range. The resulting value represents the transition probability to be assigned to the current device. To select a processing rate for the device generate a random number in the discrete closed integer range of [1,10] from a uniform distribution. This number represents the corresponding ordinal element in the relative processing rate subset that is to be assigned to the device.

The assignment algorithm is :

```
BEGIN ASSIGN(p_SPR)
        prob = 1.0 - p_SPR
        STEP BY 1  j=1   TO   s
        DO
                p_j = .75*(prob)*Ranc[0,1]
                IF   j = 1
                        THEN  u_j = 1.0
                        ELSE  u_j = speed(Rand[1,10])
                prob = prob - p_j
        END
        p_s = p_s + prob
END ASSIGN
```

where

Ranc[a,b] is a function which generates a uniformly distributed random number in the continuous, open interval from a to b, and

Rand[1,n] is a function which genarates a uniformly distributed random integer in the discrete, closed interval from 1 to n.

We have now converted from an infinitely large population space to a reasonably sized sample space of 1800 combinations. The resulting sample space parameter values are:

$R = \{1, 2, 5, 8\}$

$J = \{2, 4, 6\}$

$s = \{2, 6, 11\}$

$u_{SPR} = \{.01, .02, .05, .1, .2, .5, 1., 2., 5., 10.\}$

$u_{CPU} = 1.$

u_j = a random selection from the same set as u_{SPR}, $j=2, \ldots, s$

$p_{SPR} = \{.1, .25, .5, .75, .9\}$

p_j = a random selection from the same set as p_{SPR}, $j=1, \ldots, s$

Using our Fortran implementation of both models, values for the throughput and mean queue length of both the CPU and SPR have been generated in the following manner. For each of the 15 combinations of $s \times p_{SPR}$, 15 corresponding groups of values for p_j and u_j were generated. A program, based on the ASSIGN algorithm above, was constructed in Fortran to do this, and its results are listed in appendix D. The entire 120 combinations of $u_{SPR} \times R \times J$ were used 15 times, once for each of the 15 groups of transition probabilities and relative processing rates.

Table IV-1 contains the accounting statistics on the actual CPU processing times for both the exact and approximate SCS models executed on a DEC 10 computer. Both models were executed in a batch environment with all input data completely specified in advance in a file. The times are based on an execution unit which computes a set of 40 data points. This represents one value from the J set, one group of transition probability and relative processing rate values, and the entire 40 combinations of the $u_{SPR} \times R$ set.

The processing time for an execution unit of the exact SCS model should be $10\ \Delta t\ O[(J+1)^1 + (J+1)^2 + (J+1)^5 + (J+1)^8]$, where Δt is the average time per operation. The high order term dominates the expression, which may be approximated by $10\ \Delta t\ O[(J+1)^8]$. From this the expected relative processing time of an execution unit is $T_b\ (J+1)^8 / (J_b+1)^8$, where

40 Data Point Execution Unit	Exact SCS Model	Approximate SCS Model
J=2 CPU time Elapsed time	 1.4 Min. 2.2 Min.	 2.9 Sec. 3.5 Sec.
J=4 CPU time Elapsed time	 13.0 Min. 14.7 Min.	 2.9 Sec. 3.5 Sec.
J=6 CPU time Elapsed time	 148 Min. 178 Min.	 2.9 Sec. 3.5 Sec.
TOTAL for 1800 points CPU time Elapsed time	 2436 Min. = 40.6 Hrs. 2924 Min. = 48.8 Hrs.	 131 Sec. 158 Sec.

Table IV-1. Execution unit processing times.

J_b is a base reference value of jobs per ICS and T_b is the corresponding average measured processing time. For the sample, $J_b = 2$ and T_b is approximately 1.4 minutes. Therefore, for an execution unit of $J = 6$ the increase in processing time is $(6+1)^8/(2+1)^8 = (7/3)^8 \simeq 878$ times longer than the $J_b = 2$ execution unit, or 1229 minutes. Since the system is a balanced one (all ICSs are identical), the actual computations need only be carried out for one ICS, rather than for all R. This reduces the number of computations by approximately 1/R, resulting in a revised increased processing time of 109 (vs. 878) times the $J_b = 2$ excution unit, or 152 minutes. This agrees reasonably well with the average measured value of 148 minutes. Similarly for the $J = 4$ execution unit an increase of about 7.5 times is predicted, or 10.5 minutes compared with the average measured value of 13.0 minutes. These measurements verify the relations developed in chapter III for the number of operations required to compute the performance measures for any given SCS system configuration. The majority of error is attributed to approximating this relationship by only its dominant term.

In contrast to these exponentially increasing processing times on the order of minutes and hours, the processing times of our approximate SCS model are on the order of seconds, and for balanced systems are independent of R and J. This is verified by the average measured processing times in table IV-1, and by examination of (7) and the BIN algorithm used for its solution.

Tables IV-2 through IV-5 contain relative error (= {exact value - approximate value}/exact value) statistics produced from the results of computing the performance measures from all the sample space parameters for both models. These error statistics consist of mean, variance, minimum, and maximum values for each of the individual parameters and for all the parameters together. Each of the relative error values are further organized as a function of ρ_{SPR}, the traffic intensity of the SPR. The relative error statistics are listed in pairs, first all values of ρ_{SPR} and second $\rho_{SPR} < .9$. Each table consists of 6 subtables. The first (top) subtable contains the overall statistics for the indicated performance measure. The heading of the first column of each of the remaining 5 subtables indicates the parameter being investigated within that subtable. Each row of a subtable represents the statistics for a single value of the parameter being investigated, with all other parameters varied through their complete sample-space ranges. The first column contains the value of the parameter, the second column contains the number of data points used to compute the statistics, and the remaining columns contain the statistics as indicated.

ρ	Points	Mean	Variance	Minimum	Maximum
—	1650/1158	0.0904/ 0.1171	0.0189/ 0.0221	-0.4489/-0.3824	0.3636/ 0.3636

p_{SPR}	Points	Mean	Variance	Minimum	Maximum
0.10	330/ 266	0.1030/ 0.1194	0.0139/ 0.0151	-0.2649/-0.1949	0.3396/ 0.3396
0.25	330/ 260	0.1237/ 0.1506	0.0199/ 0.0205	-0.3698/-0.3062	0.3529/ 0.3529
0.50	330/ 208	0.0139/ 0.0049	0.0211/ 0.0272	-0.3648/-0.3017	0.3500/ 0.3500
0.75	330/ 227	0.1590/ 0.2124	0.0103/ 0.0040	0.0000/ 0.0330	0.3382/ 0.3382
0.90	330/ 197	0.0525/ 0.0781	0.0162/ 0.0224	-0.4489/-0.3824	0.3636/ 0.3636

J	Points	Mean	Variance	Minimum	Maximum
2.00	600/ 458	0.1119/ 0.1308	0.0243/ 0.0286	-0.3824/-0.3824	0.3636/ 0.3636
4.00	600/ 411	0.0801/ 0.1132	0.0173/ 0.0193	-0.4489/-0.3062	0.3333/ 0.2619
6.00	450/ 289	0.0754/ 0.1009	0.0128/ 0.0152	-0.3698/-0.1949	0.2330/ 0.2330

s	Points	Mean	Variance	Minimum	Maximum
2.00	550/ 387	0.0808/ 0.1048	0.0209/ 0.0250	-0.4489/-0.3824	0.3636/ 0.3636
6.00	550/ 389	0.1180/ 0.1526	0.0140/ 0.0140	-0.3028/-0.2454	0.3636/ 0.3636
11.00	550/ 382	0.0724/ 0.0933	0.0205/ 0.0255	-0.3698/-0.3062	0.3636/ 0.3636

R	Points	Mean	Variance	Minimum	Maximum
1.00	450/ 450	0.1446/ 0.1446	0.0206/ 0.0206	-0.1744/-0.1744	0.3636/ 0.3636
2.00	450/ 383	0.1059/ 0.1100	0.0155/ 0.0180	-0.1765/-0.1765	0.3429/ 0.3429
5.00	450/ 213	0.0512/ 0.0846	0.0152/ 0.0247	-0.3698/-0.3062	0.3333/ 0.3333
8.00	300/ 112	0.0448/ 0.0923	0.0178/ 0.0318	-0.4489/-0.3824	0.3429/ 0.3429

u_{SPR}	Points	Mean	Variance	Minimum	Maximum
0.01	165/ 76	0.1312/ 0.2082	0.0138/ 0.0065	0.0000/ 0.0909	0.3636/ 0.3636
0.02	165/ 79	0.1072/ 0.1925	0.0104/ 0.0059	0.0000/ 0.0769	0.3529/ 0.3529
0.05	165/ 85	0.1178/ 0.1916	0.0090/ 0.0051	0.0000/ 0.0857	0.3458/ 0.3458
0.10	165/ 92	0.1207/ 0.1820	0.0086/ 0.0065	-0.0295/-0.0295	0.3471/ 0.3471
0.20	165/ 100	0.1158/ 0.1662	0.0099/ 0.0091	-0.1424/-0.1424	0.3333/ 0.3333
0.50	165/ 120	0.0849/ 0.1127	0.0148/ 0.0166	-0.2649/-0.1949	0.3333/ 0.3333
1.00	165/ 131	0.0543/ 0.0760	0.0235/ 0.0259	-0.3698/-0.3062	0.3333/ 0.3333
2.00	165/ 147	0.0423/ 0.0627	0.0322/ 0.0312	-0.3648/-0.3017	0.3333/ 0.3333
5.00	165/ 163	0.0579/ 0.0633	0.0317/ 0.0296	-0.4489/-0.3824	0.3333/ 0.3333
10.00	165/ 165	0.0719/ 0.0719	0.0264/ 0.0264	-0.1689/-0.1689	0.3333/ 0.3333

Table IV-2. Relative Error statistics for Q_{CPU}, for both all $\rho/\rho < .90$.

ρ	Points	Mean	Variance	Minimum	Maximum
—	1650/1158	0.1551/ 0.1968	0.0066/ 0.0034	0.0000/ 0.0983	0.3383/ 0.3383

p_{SPR}	Points	Mean	Variance	Minimum	Maximum
0.10	330/ 266	0.1627/ 0.1879	0.0039/ 0.0015	0.0300/ 0.1025	0.3347/ 0.3347
0.25	330/ 260	0.1782/ 0.2110	0.0074/ 0.0042	0.0300/ 0.1000	0.3371/ 0.3371
0.50	330/ 208	0.1388/ 0.1868	0.0060/ 0.0031	0.0250/ 0.0990	0.3367/ 0.3367
0.75	330/ 227	0.1592/ 0.2055	0.0078/ 0.0041	0.0189/ 0.0983	0.3383/ 0.3383
0.90	330/ 197	0.1365/ 0.1905	0.0069/ 0.0039	0.0000/ 0.1005	0.3378/ 0.3378

J	Points	Mean	Variance	Minimum	Maximum
2.00	600/ 458	0.2026/ 0.2418	0.0079/ 0.0038	0.0400/ 0.0992	0.3383/ 0.3383
4.00	600/ 411	0.1378/ 0.1804	0.0048/ 0.0011	0.0241/ 0.0983	0.2156/ 0.2156
6.00	450/ 289	0.1147/ 0.1486	0.0024/ 0.0002	0.0000/ 0.0990	0.1642/ 0.1642

s	Points	Mean	Variance	Minimum	Maximum
2.00	550/ 387	0.1611/ 0.2051	0.0080/ 0.0046	0.0000/ 0.0988	0.3383/ 0.3383
6.00	550/ 389	0.1527/ 0.1922	0.0056/ 0.0024	0.0000/ 0.0992	0.3383/ 0.3383
11.00	550/ 382	0.1513/ 0.1930	0.0063/ 0.0031	0.0000/ 0.0983	0.3378/ 0.3378

R	Points	Mean	Variance	Minimum	Maximum
1.00	450/ 450	0.2068/ 0.2068	0.0037/ 0.0037	0.1169/ 0.1169	0.3383/ 0.3383
2.00	450/ 383	0.1687/ 0.1840	0.0037/ 0.0028	0.0721/ 0.0990	0.3362/ 0.3362
5.00	450/ 213	0.1180/ 0.1880	0.0063/ 0.0033	0.0000/ 0.0983	0.3371/ 0.3371
8.00	300/ 112	0.1125/ 0.2166	0.0078/ 0.0030	0.0241/ 0.0988	0.3371/ 0.3371

u_{SPR}	Points	Mean	Variance	Minimum	Maximum
0.01	165/ 76	0.1238/ 0.1977	0.0074/ 0.0052	0.0000/ 0.1000	0.3383/ 0.3383
0.02	165/ 79	0.1216/ 0.1934	0.0075/ 0.0051	0.0189/ 0.1025	0.3378/ 0.3378
0.05	165/ 85	0.1259/ 0.1920	0.0073/ 0.0046	0.0241/ 0.1025	0.3376/ 0.3376
0.10	165/ 92	0.1316/ 0.1908	0.0069/ 0.0041	0.0288/ 0.1011	0.3362/ 0.3362
0.20	165/ 100	0.1377/ 0.1899	0.0062/ 0.0030	0.0300/ 0.0994	0.3371/ 0.3371
0.50	165/ 120	0.1474/ 0.1827	0.0054/ 0.0027	0.0303/ 0.0983	0.3362/ 0.3362
1.00	165/ 131	0.1618/ 0.1896	0.0053/ 0.0028	0.0317/ 0.0992	0.3352/ 0.3352
2.00	165/ 147	0.1835/ 0.1986	0.0044/ 0.0028	0.0335/ 0.0988	0.3352/ 0.3352
5.00	165/ 163	0.2061/ 0.2078	0.0030/ 0.0028	0.0562/ 0.1387	0.3352/ 0.3352
10.00	165/ 165	0.2113/ 0.2113	0.0029/ 0.0029	0.1387/ 0.1387	0.3359/ 0.3359

Table IV-3. Relative Error statistics for T_{CPU}, for both all $\rho/\rho < .90$.

ρ	Points	Mean	Variance	Minimum	Maximum
—	1650/1158	0.1042/ 0.1459	0.0195/ 0.0215	-0.1595/-0.1595	0.5792/ 0.5792

p_{SPR}	Points	Mean	Variance	Minimum	Maximum
0.10	330/ 266	0.1323/ 0.1623	0.0160/ 0.0148	-0.1595/-0.1595	0.5000/ 0.5000
0.25	330/ 260	0.1532/ 0.1920	0.0232/ 0.0220	-0.1229/-0.1229	0.5401/ 0.5401
0.50	330/ 208	0.0659/ 0.1013	0.0138/ 0.0178	-0.1529/-0.1529	0.4651/ 0.4651
0.75	330/ 227	0.1095/ 0.1566	0.0219/ 0.0244	-0.1205/-0.1205	0.5792/ 0.5792
0.90	330/ 197	0.0603/ 0.0980	0.0161/ 0.0230	-0.1507/-0.1507	0.5449/ 0.5449

J	Points	Mean	Variance	Minimum	Maximum
2.00	600/ 458	0.1159/ 0.1519	0.0253/ 0.0277	-0.1595/-0.1595	0.5449/ 0.5449
4.00	600/ 411	0.1049/ 0.1483	0.0187/ 0.0206	-0.1291/-0.1291	0.5792/ 0.5792
6.00	450/ 289	0.0879/ 0.1332	0.0124/ 0.0127	-0.1216/-0.1216	0.5091/ 0.5091

s	Points	Mean	Variance	Minimum	Maximum
2.00	550/ 387	0.1188/ 0.1658	0.0227/ 0.0243	-0.1338/-0.1338	0.5449/ 0.5449
6.00	550/ 389	0.0993/ 0.1386	0.0175/ 0.0192	-0.1595/-0.1595	0.5000/ 0.5000
11.00	550/ 382	0.0946/ 0.1333	0.0180/ 0.0204	-0.1594/-0.1594	0.5792/ 0.5792

R	Points	Mean	Variance	Minimum	Maximum
1.00	450/ 450	0.0895/ 0.0895	0.0180/ 0.0180	-0.1595/-0.1595	0.5000/ 0.5000
2.00	450/ 383	0.1037/ 0.1227	0.0168/ 0.0173	-0.1177/-0.1177	0.5000/ 0.5000
5.00	450/ 213	0.1097/ 0.2240	0.0196/ 0.0154	-0.0273/-0.0273	0.5091/ 0.5091
8.00	300/ 112	0.1190/ 0.3039	0.0251/ 0.0104	-0.0056/ 0.0645	0.5792/ 0.5792

u_{SPR}	Points	Mean	Variance	Minimum	Maximum
0.01	165/ 76	-0.0054/-0.0109	0.0005/ 0.0011	-0.1595/-0.1595	0.0901/ 0.0901
0.02	165/ 79	-0.0003/-0.0001	0.0033/ 0.0069	-0.1594/-0.1594	0.3029/ 0.3029
0.05	165/ 85	0.0178/ 0.0325	0.0102/ 0.0188	-0.1338/-0.1338	0.5026/ 0.5026
0.10	165/ 92	0.0321/ 0.0570	0.0126/ 0.0212	-0.1229/-0.1229	0.5379/ 0.5379
0.20	165/ 100	0.0521/ 0.0832	0.0164/ 0.0241	-0.1529/-0.1529	0.4294/ 0.4294
0.50	165/ 120	0.0975/ 0.1316	0.0198/ 0.0224	-0.1507/-0.1507	0.4998/ 0.4998
1.00	165/ 131	0.1548/ 0.1888	0.0156/ 0.0134	-0.0870/-0.0870	0.5792/ 0.5792
2.00	165/ 147	0.2159/ 0.2351	0.0104/ 0.0077	-0.0004/-0.0004	0.5401/ 0.5401
5.00	165/ 163	0.2514/ 0.2517	0.0077/ 0.0078	0.0000/ 0.0000	0.5091/ 0.5091
10.00	165/ 165	0.2267/ 0.2267	0.0088/ 0.0088	0.0000/ 0.0000	0.5449/ 0.5449

Table IV-4. Relative Error statistics for Q_{SPR}, for both all $\rho/\rho < .90$.

ρ	Points	Mean	Variance	Minimum	Maximum
—	1650/1158	0.1549/ 0.1967	0.0067/ 0.0034	0.0300/ 0.0988	0.3409/ 0.3409

p_{SPR}	Points	Mean	Variance	Minimum	Maximum
0.10	330/ 266	0.1627/ 0.1879	0.0040/ 0.0015	0.0300/ 0.1000	0.3367/ 0.3367
0.25	330/ 260	0.1782/ 0.2112	0.0075/ 0.0042	0.0300/ 0.1000	0.3409/ 0.3409
0.50	330/ 208	0.1387/ 0.1868	0.0061/ 0.0031	0.0300/ 0.0990	0.3400/ 0.3400
0.75	330/ 227	0.1586/ 0.2052	0.0078/ 0.0042	0.0300/ 0.0988	0.3400/ 0.3400
0.90	330/ 197	0.1361/ 0.1900	0.0069/ 0.0039	0.0300/ 0.1006	0.3400/ 0.3400

J	Points	Mean	Variance	Minimum	Maximum
2.00	600/ 458	0.2025/ 0.2419	0.0080/ 0.0038	0.0590/ 0.0989	0.3409/ 0.3409
4.00	600/ 411	0.1371/ 0.1802	0.0049/ 0.0012	0.0300/ 0.0988	0.2145/ 0.2145
6.00	450/ 289	0.1150/ 0.1486	0.0023/ 0.0002	0.0300/ 0.0990	0.1659/ 0.1659

s	Points	Mean	Variance	Minimum	Maximum
2.00	550/ 387	0.1611/ 0.2052	0.0080/ 0.0046	0.0300/ 0.0989	0.3409/ 0.3409
6.00	550/ 389	0.1525/ 0.1920	0.0057/ 0.0025	0.0300/ 0.0989	0.3400/ 0.3400
11.00	550/ 382	0.1511/ 0.1928	0.0063/ 0.0031	0.0300/ 0.0988	0.3400/ 0.3400

R	Points	Mean	Variance	Minimum	Maximum
1.00	450/ 450	0.2070/ 0.2070	0.0037/ 0.0037	0.1175/ 0.1175	0.3409/ 0.3409
2.00	450/ 383	0.1684/ 0.1836	0.0037/ 0.0028	0.0750/ 0.0990	0.3371/ 0.3371
5.00	450/ 213	0.1182/ 0.1880	0.0062/ 0.0033	0.0300/ 0.0988	0.3364/ 0.3364
8.00	300/ 112	0.1115/ 0.2166	0.0079/ 0.0030	0.0300/ 0.0989	0.3357/ 0.3357

u_{SPR}	Points	Mean	Variance	Minimum	Maximum
0.01	165/ 76	0.1208/ 0.1959	0.0077/ 0.0056	0.0300/ 0.1000	0.3400/ 0.3400
0.02	165/ 79	0.1219/ 0.1936	0.0074/ 0.0051	0.0300/ 0.1000	0.3400/ 0.3400
0.05	165/ 85	0.1264/ 0.1919	0.0072/ 0.0045	0.0300/ 0.1040	0.3367/ 0.3367
0.10	165/ 92	0.1316/ 0.1906	0.0069/ 0.0041	0.0300/ 0.1000	0.3347/ 0.3347
0.20	165/ 100	0.1378/ 0.1900	0.0062/ 0.0030	0.0305/ 0.1010	0.3409/ 0.3409
0.50	165/ 120	0.1475/ 0.1828	0.0054/ 0.0027	0.0308/ 0.0988	0.3409/ 0.3409
1.00	165/ 131	0.1618/ 0.1896	0.0053/ 0.0028	0.0315/ 0.0989	0.3409/ 0.3409
2.00	165/ 147	0.1835/ 0.1987	0.0044/ 0.0028	0.0333/ 0.0989	0.3409/ 0.3409
5.00	165/ 163	0.2061/ 0.2078	0.0030/ 0.0028	0.0563/ 0.1387	0.3409/ 0.3409
10.00	165/ 165	0.2114/ 0.2114	0.0029/ 0.0029	0.1387/ 0.1387	0.3409/ 0.3409

Table IV-5. Relative Error statistics for T_{SPR}, for both all $\rho/\rho < .90$.

We would like to mention an additional limitation that has not, to our knowledge, been discussed in the literature. This limitation is related to the size limitation as it affects the precision of the specific implementation. We noticed some erratic values from the exact model were occurring for points within our sample space whose computational complexity was the highest, namely J = 6 and R = 8. We conjecture that because of the large number of floating point operations required to evaluate the exact model at these points, some combination of accumulated round-off, overflow, or underflow errors was the cause. These erratic values were not observed for a similar implementation on a CDC 6000 series machine with a 60 bit word length. The current implementation uses a DEC 10 machine with a 36 bit word length. A possible solution for this case may be to use double precision variables vs. the single precision variables used in the current implementation. The author no longer has access to the former machine, and due to the length of the computations involved was not able to pursue this any further at this time. As a result we have eliminated these 150 data points, thereby reducing our sample-space from 1800 to 1650 data points.

Figures IV-3 through IV-6 are scatter plots (left) of the relative error for each performance measure along with its corresponding mean value plot (right) for all 1650 data points in the sample space as a function of ρ_{SPR}. A scatter plot consists of the true plotting of all the points, wherever they fall --- generally scattered. Each discrete plotted point consists of a digit representing the number of actual points encompassed by it. An asterik (*) represents 10 or more points. Figures IV-7 through IV-16 are representative scatter and mean value plots of the relative error for the SPR throughput and mean queue length as a function of ρ_{SPR} for one value of each of the five parameters. Figures IV-17 through IV-20 are representative plots of the throughput and mean queue length of the CPU and SPR vs. the processing rate of the SPR, u_{SPR}, as computed by both the exact and approximate SCS models.

As can be seen from inspection of the tables and plots, the throughput relative error for both the CPU and SPR follow a fairly narrow channel centered approximately at .20 (20%) for low to moderate traffic. As the traffic intensity becomes heavy ($\rho_{SPR} > .80$) and approaches saturation ($\rho_{SPR} \simeq 1$), the relative error tends to become small. This is consistent with results obtained by Buzen [BUZEN 74] in his use of single server approximations. It should be mentioned that in applying this approximate model, if this saturation condition occurs, one immediately knows that this device is a bottleneck and is causing serious problems.

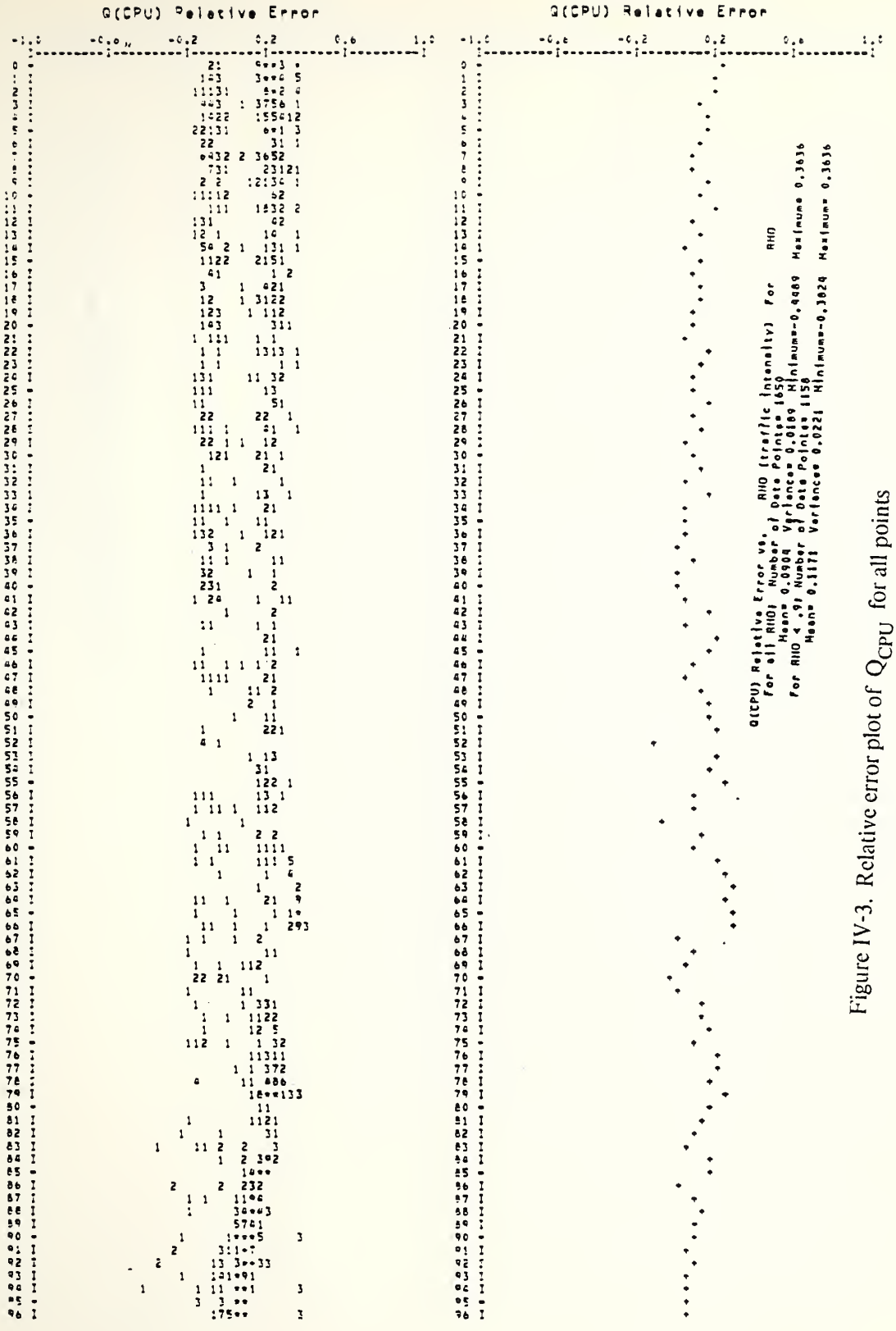

Figure IV-3. Relative error plot of Q_{CPU} for all points

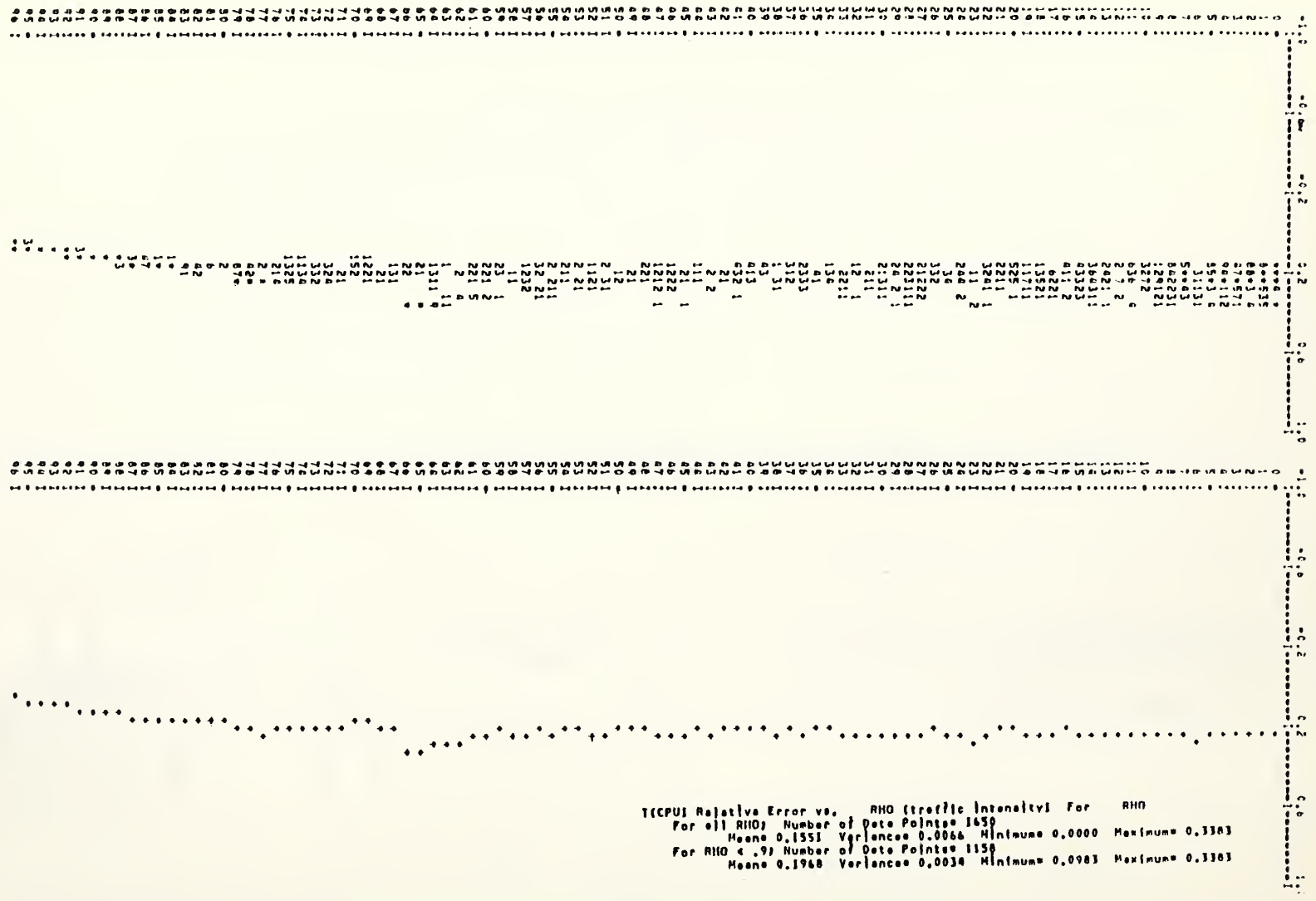

Figure IV-4. Relative error plot of T_{CPU} for all points

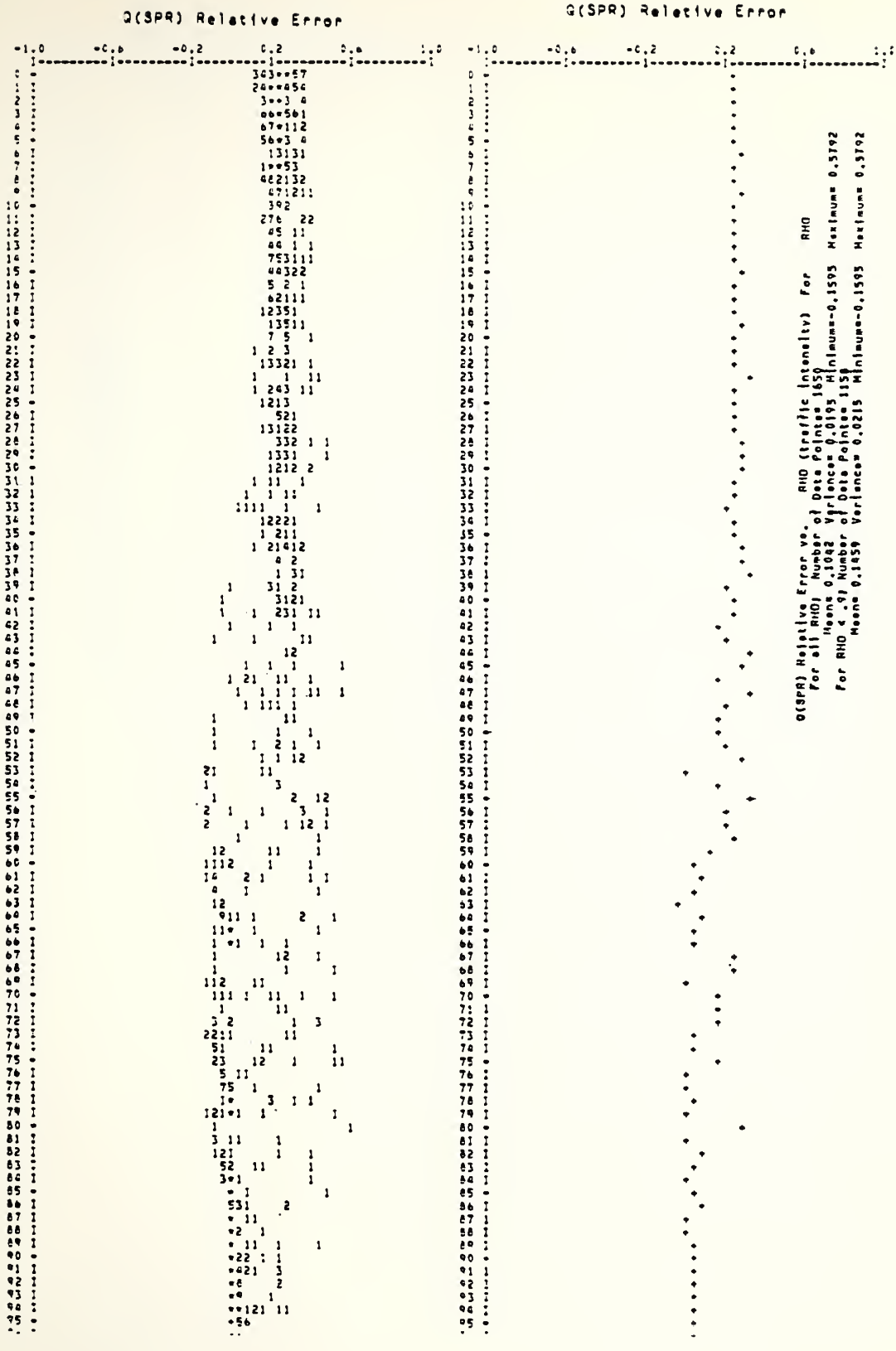

Figure IV-5. Relative error plot of Q_{SPR} for all points

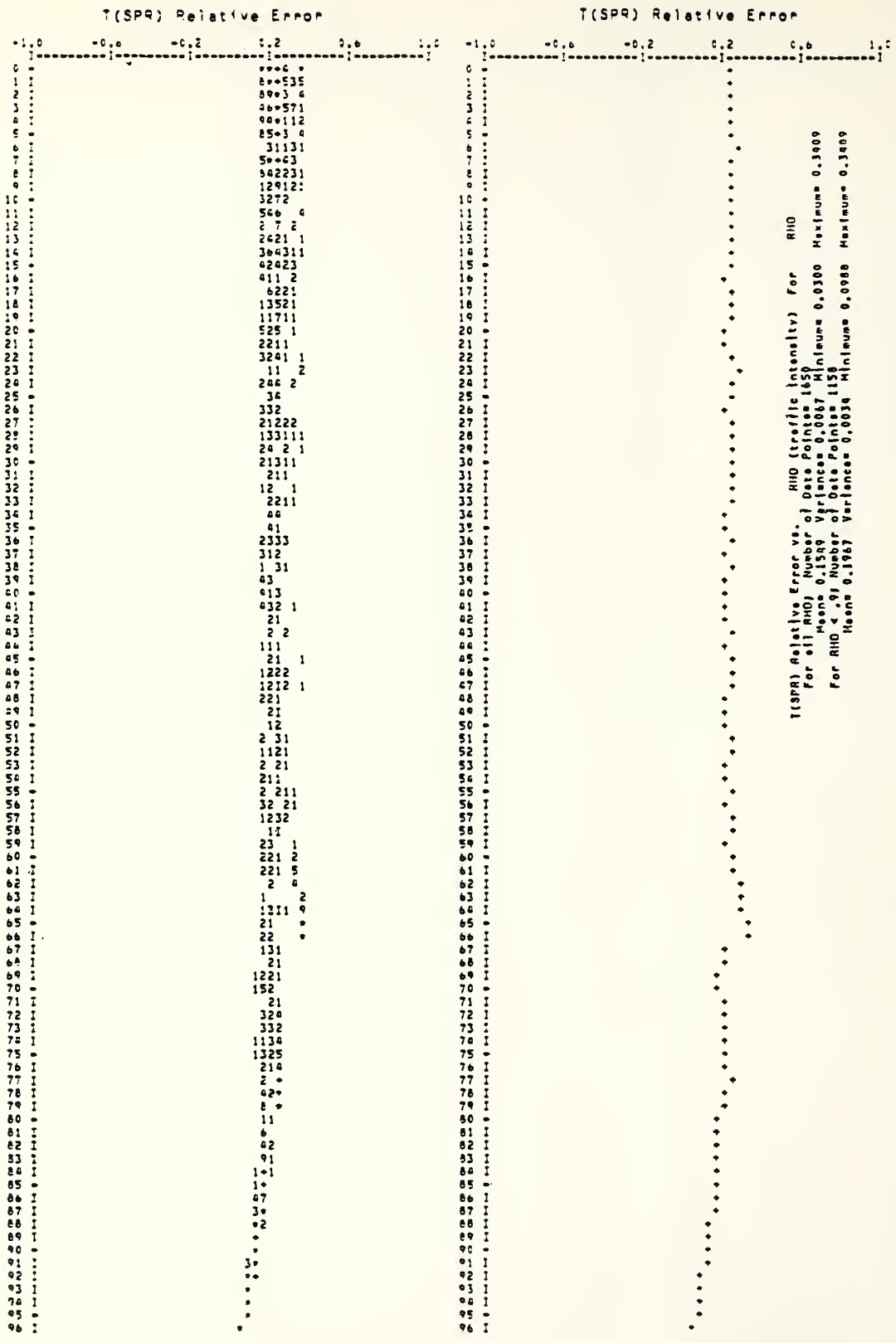

Figure IV-6. Relative error plot of T_{SPR} for all points

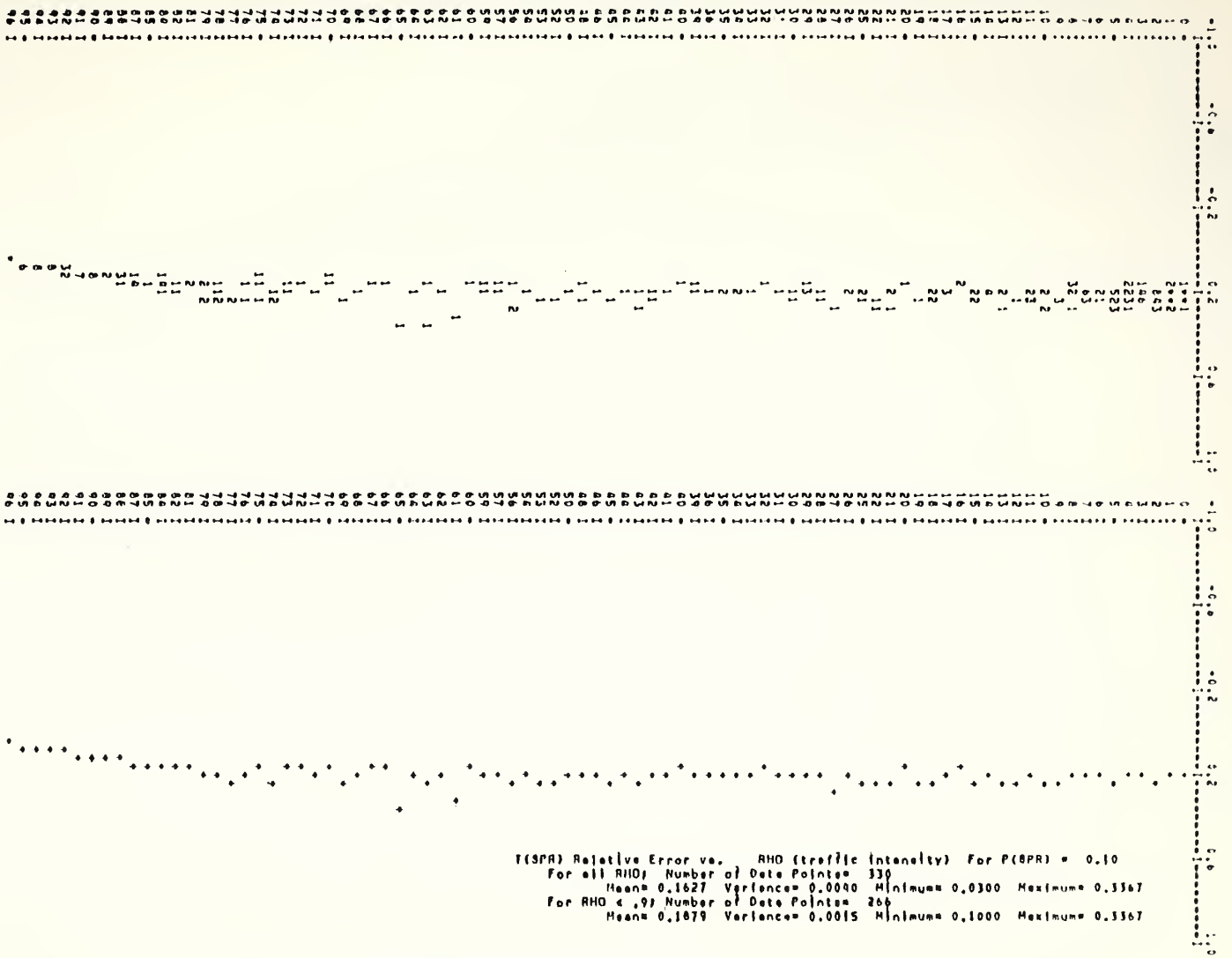

Figure IV-7. Relative error plot of T_{SPR} for P_{SPR}

Figure IV-8. Relative error plot of T_{SPR} for J

Figure IV-9. Relative error plot of T_{SPR} for s

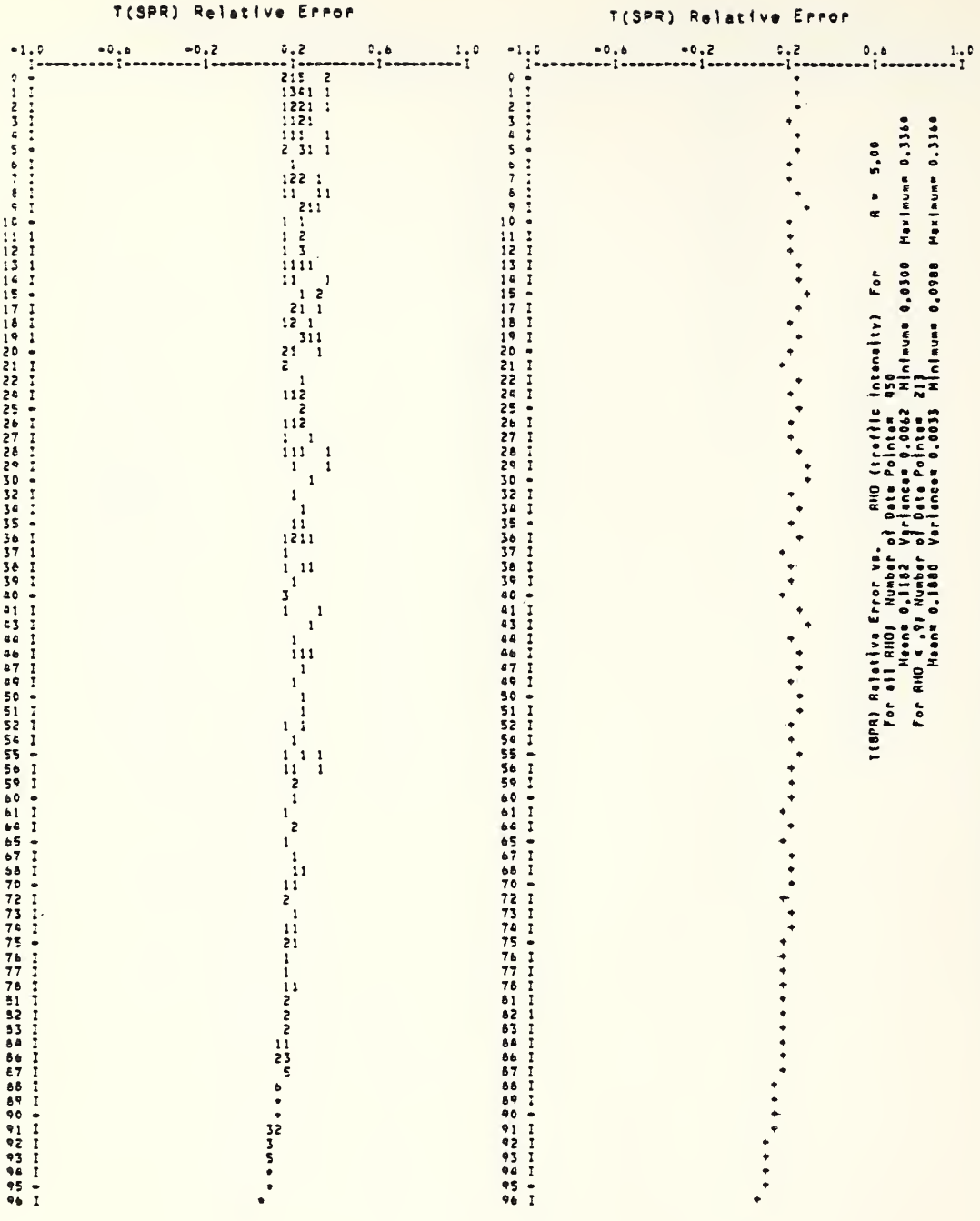

Figure IV-10. Relative error plot of T_{SPR} for R

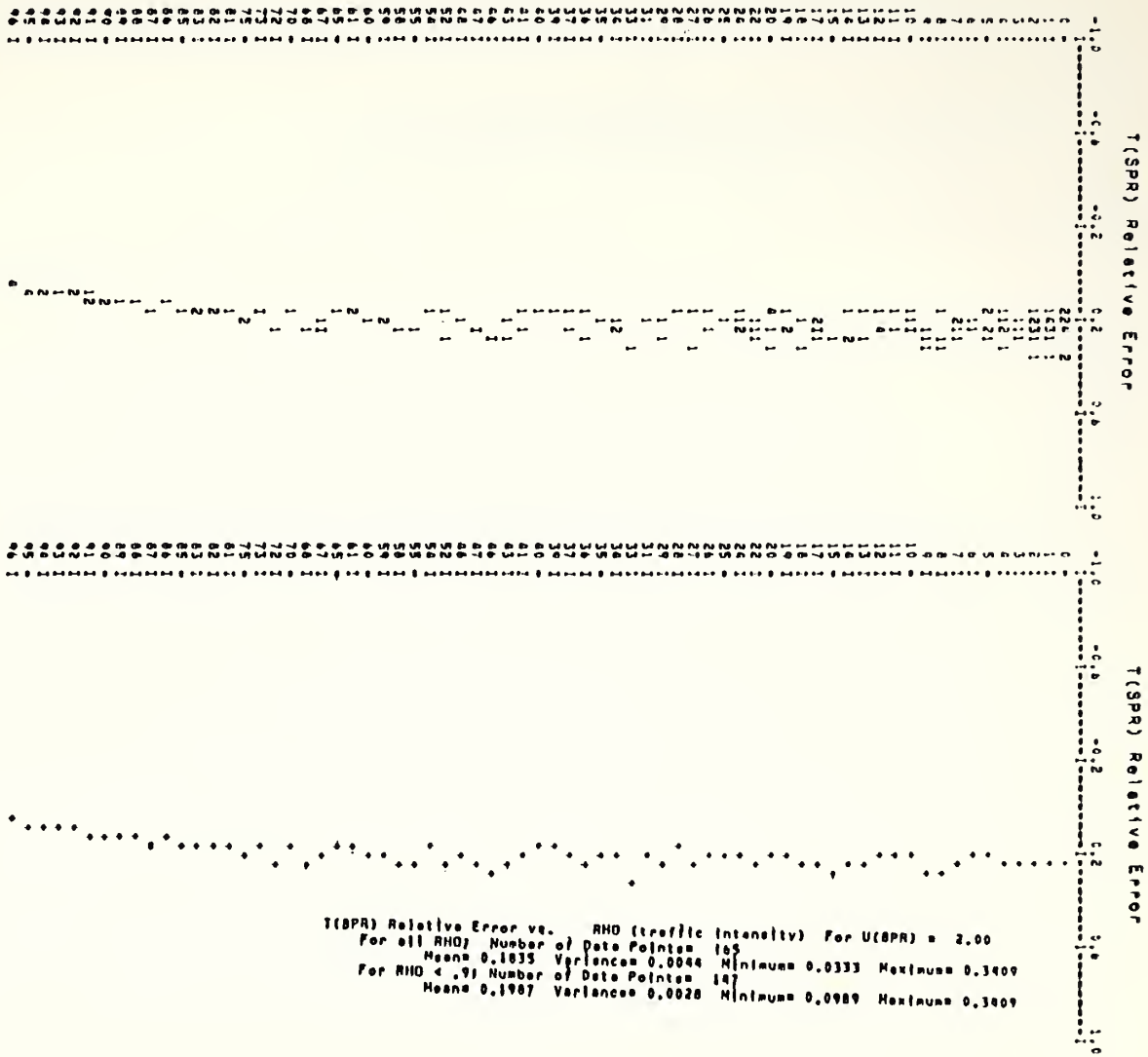

Figure IV-11. Relative error plot of T_{SPR} for u_{SPR}

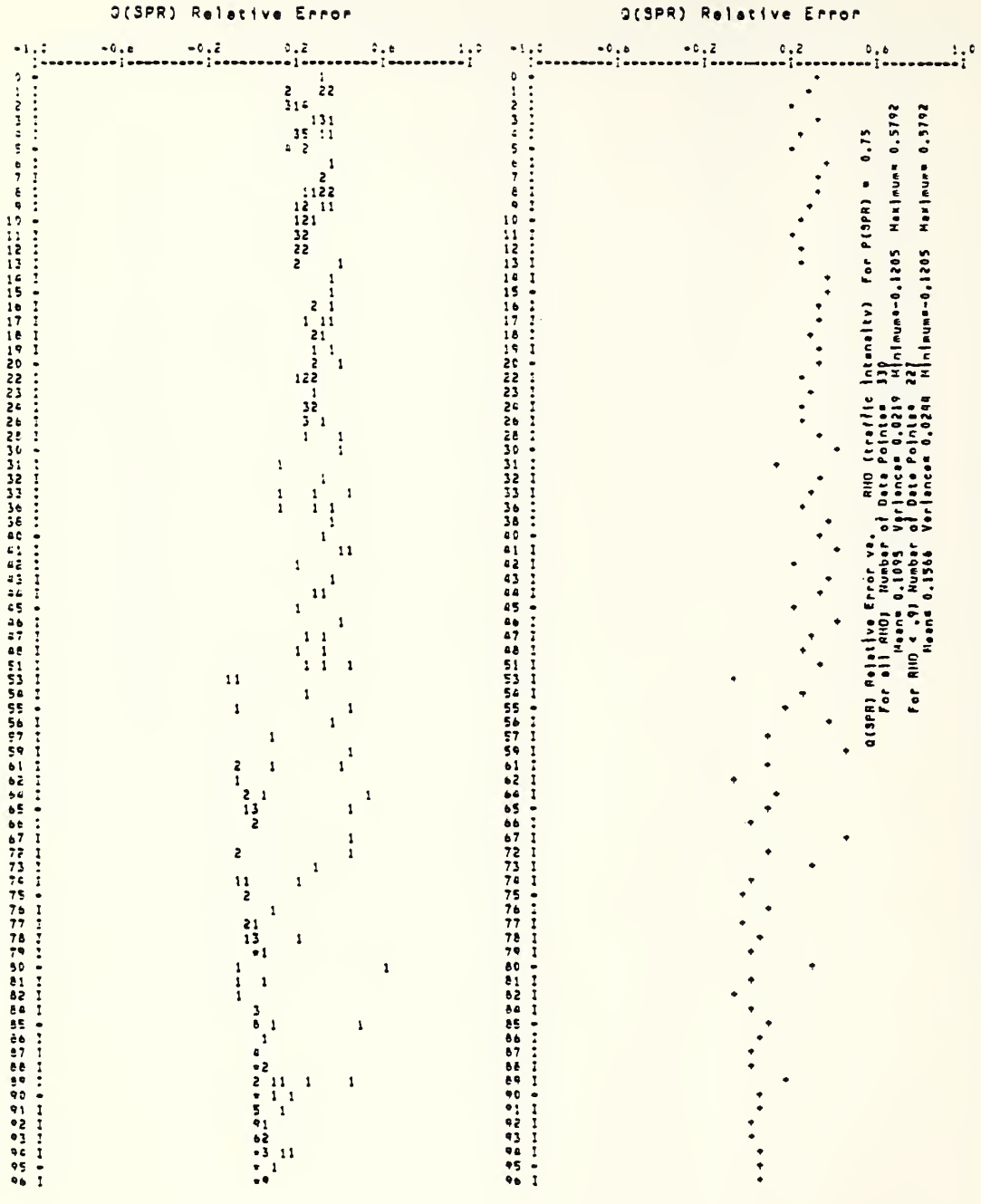

Figure IV-12. Relative error plot of Q_{SPR} for P_{SPR}

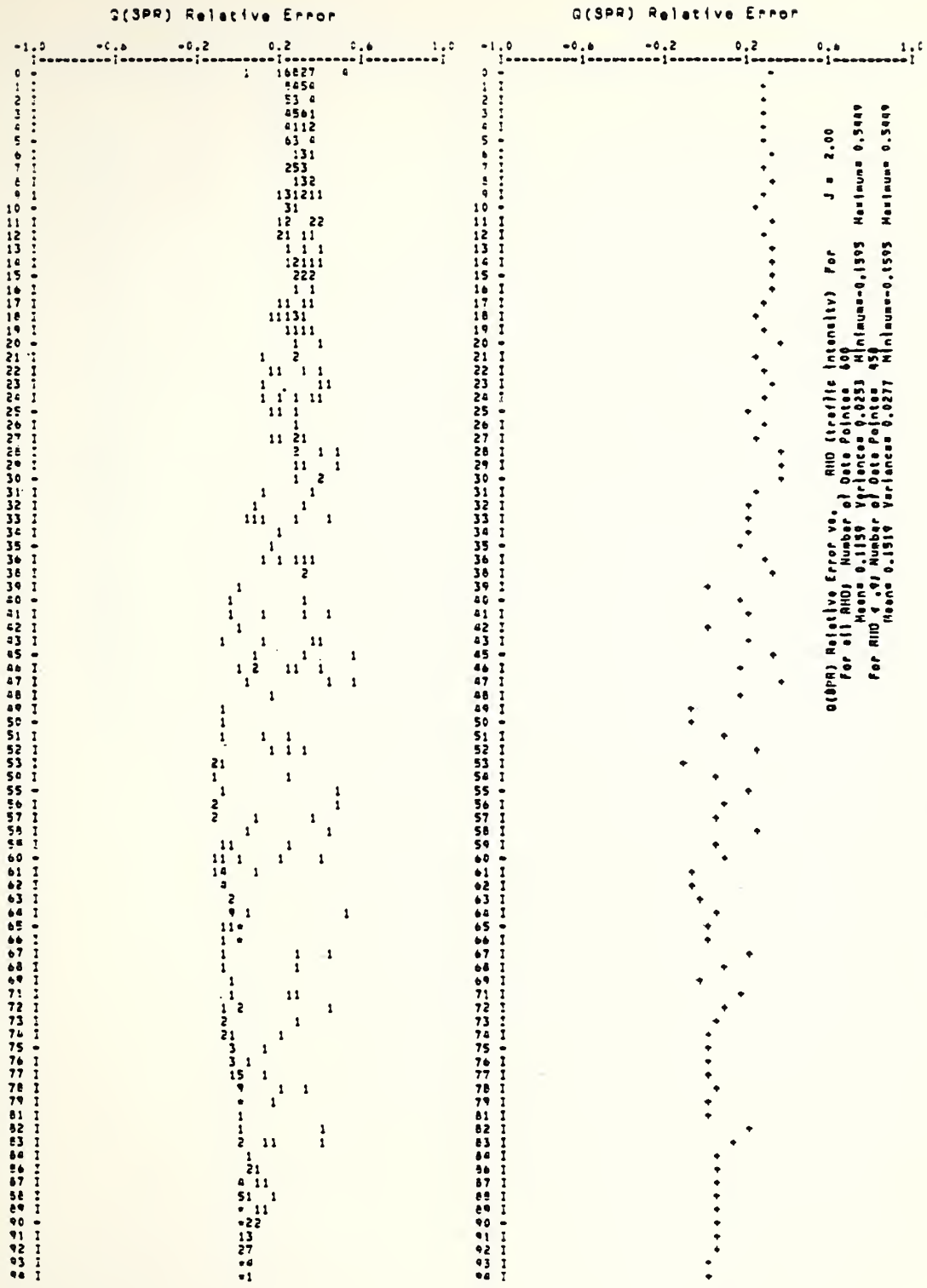

Figure IV-13. Relative error plot of Q_{SPR} for J

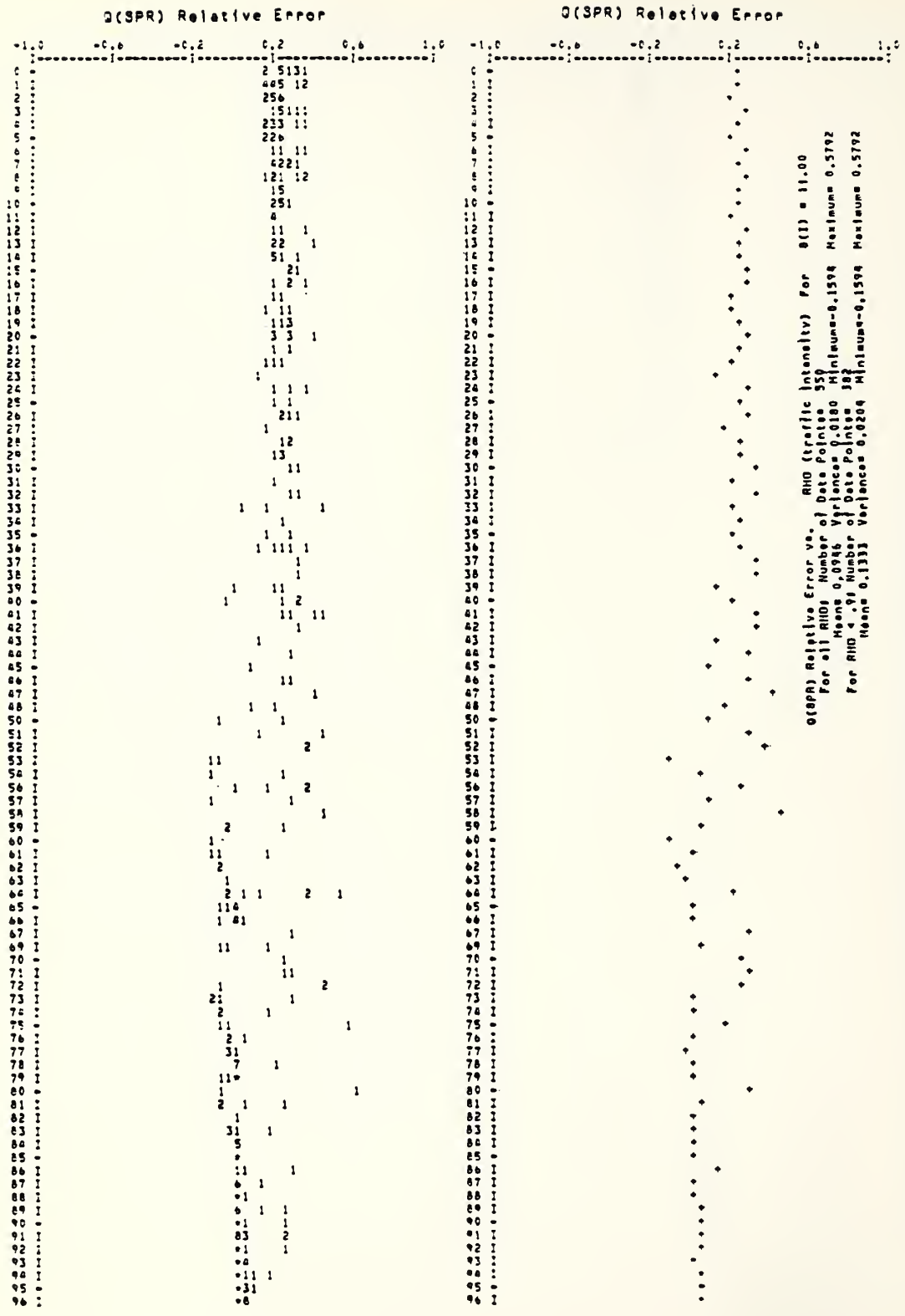

Figure IV-14. Relative error plot of Q_{SPR} for s

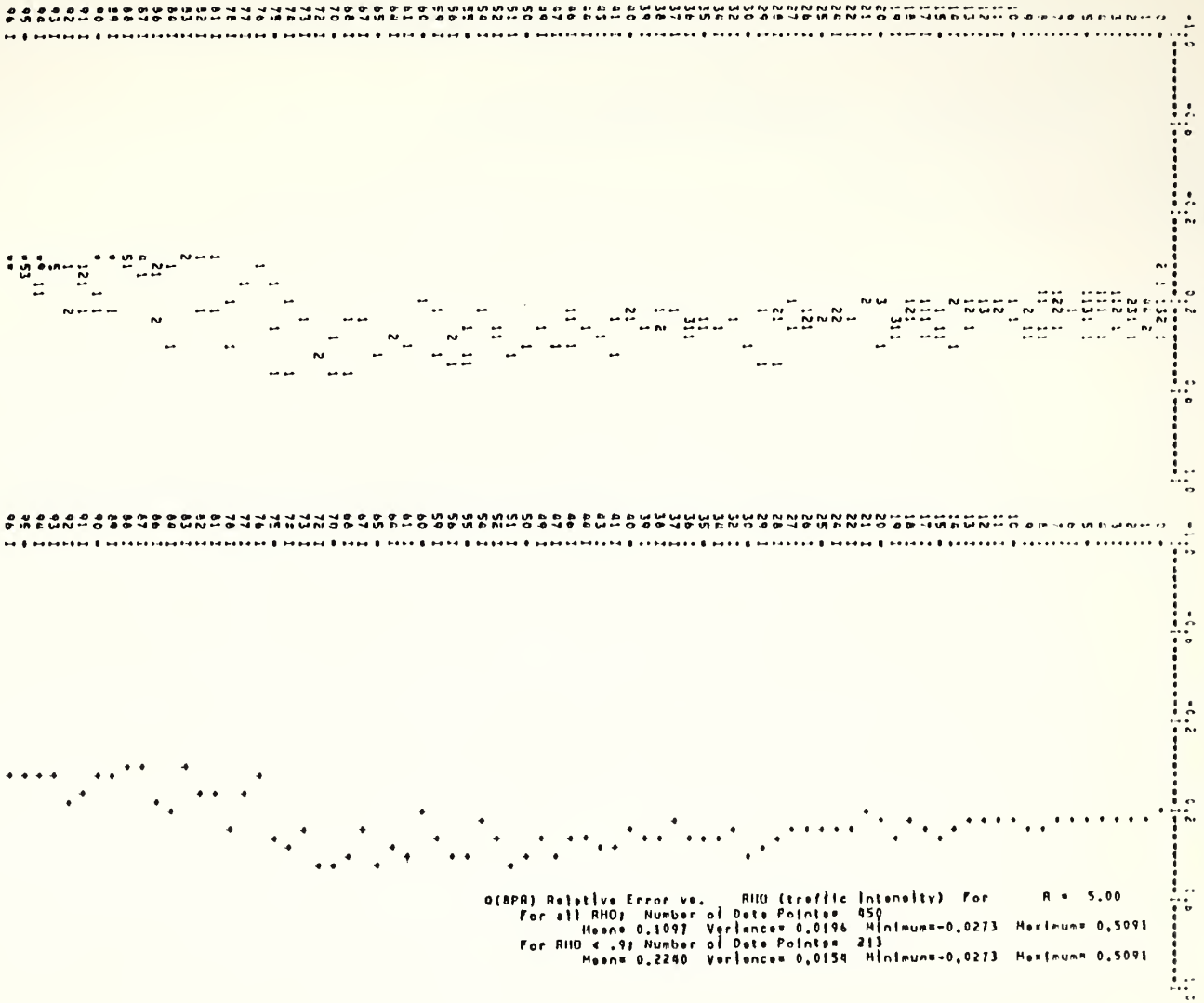

Figure IV-15. Relative error plot of Q_{SPR} for R

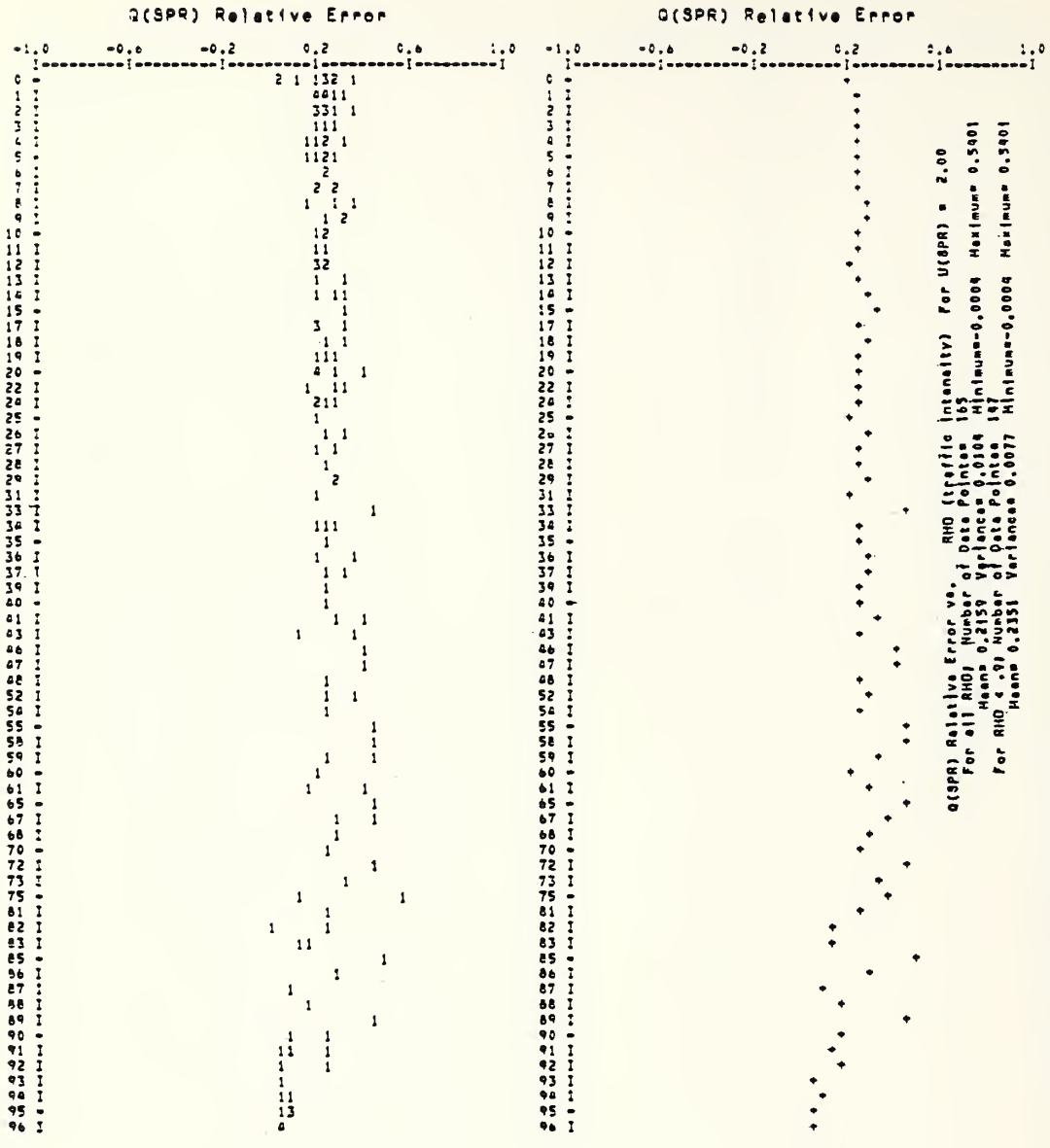

Figure IV-16. Relative error plot of Q_{SPR} for u_{SPR}

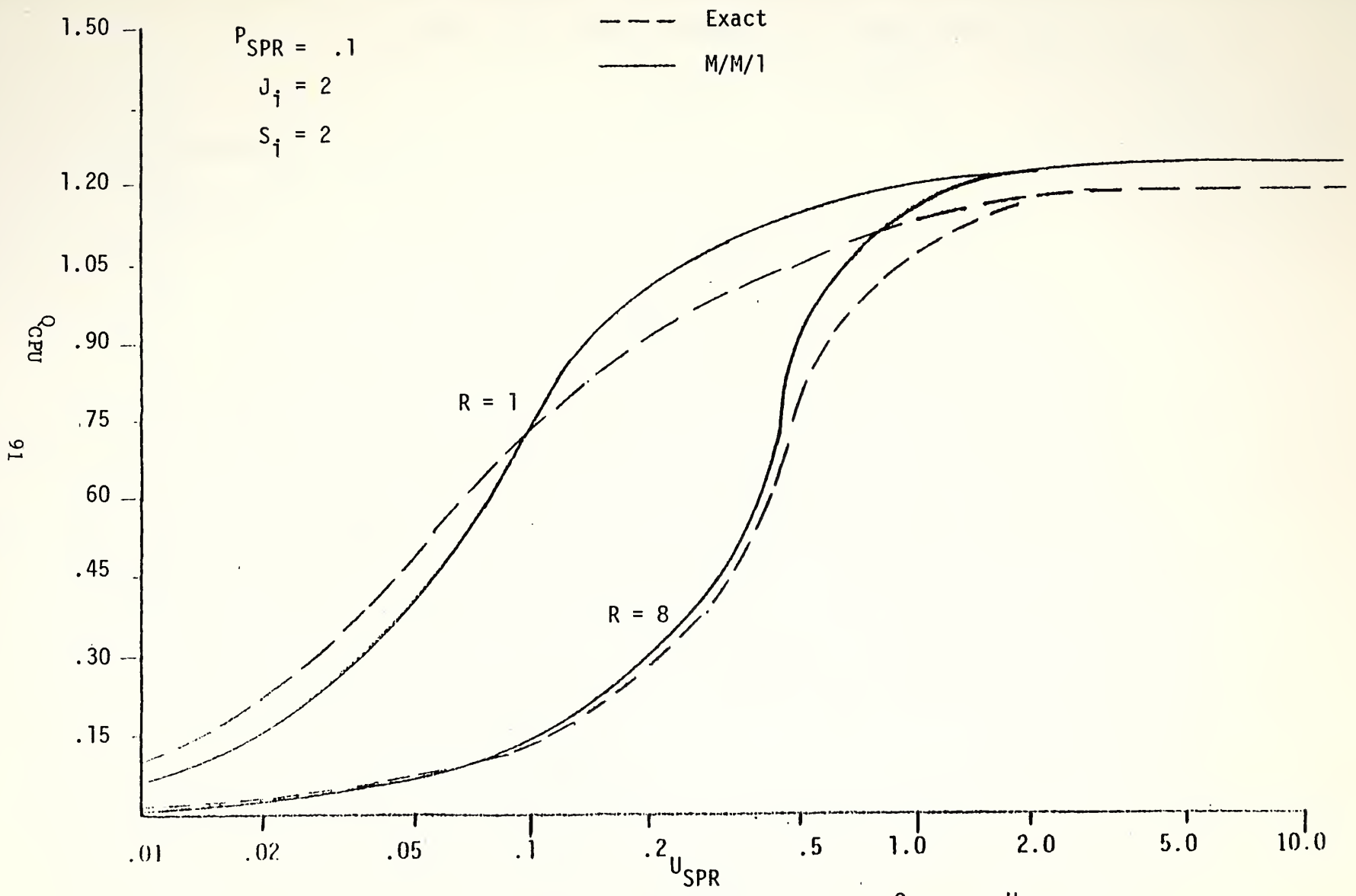

Figure IV-17. Exact and M/M/1 values for Q_{CPU} vs. U_{SPR}.

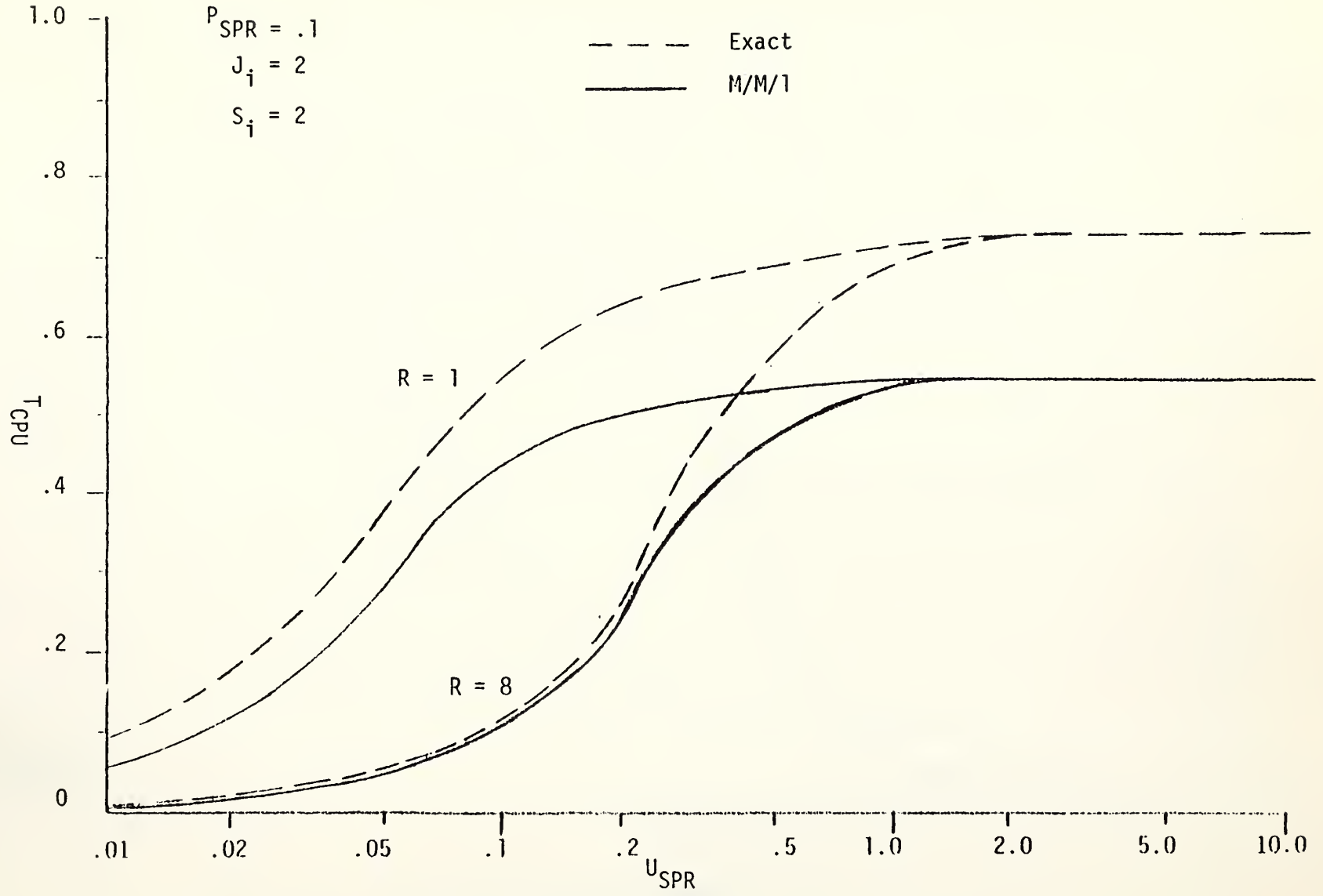

Figure IV-18. Exact and M/M/1 values for T_{CPU} vs. U_{SPR}.

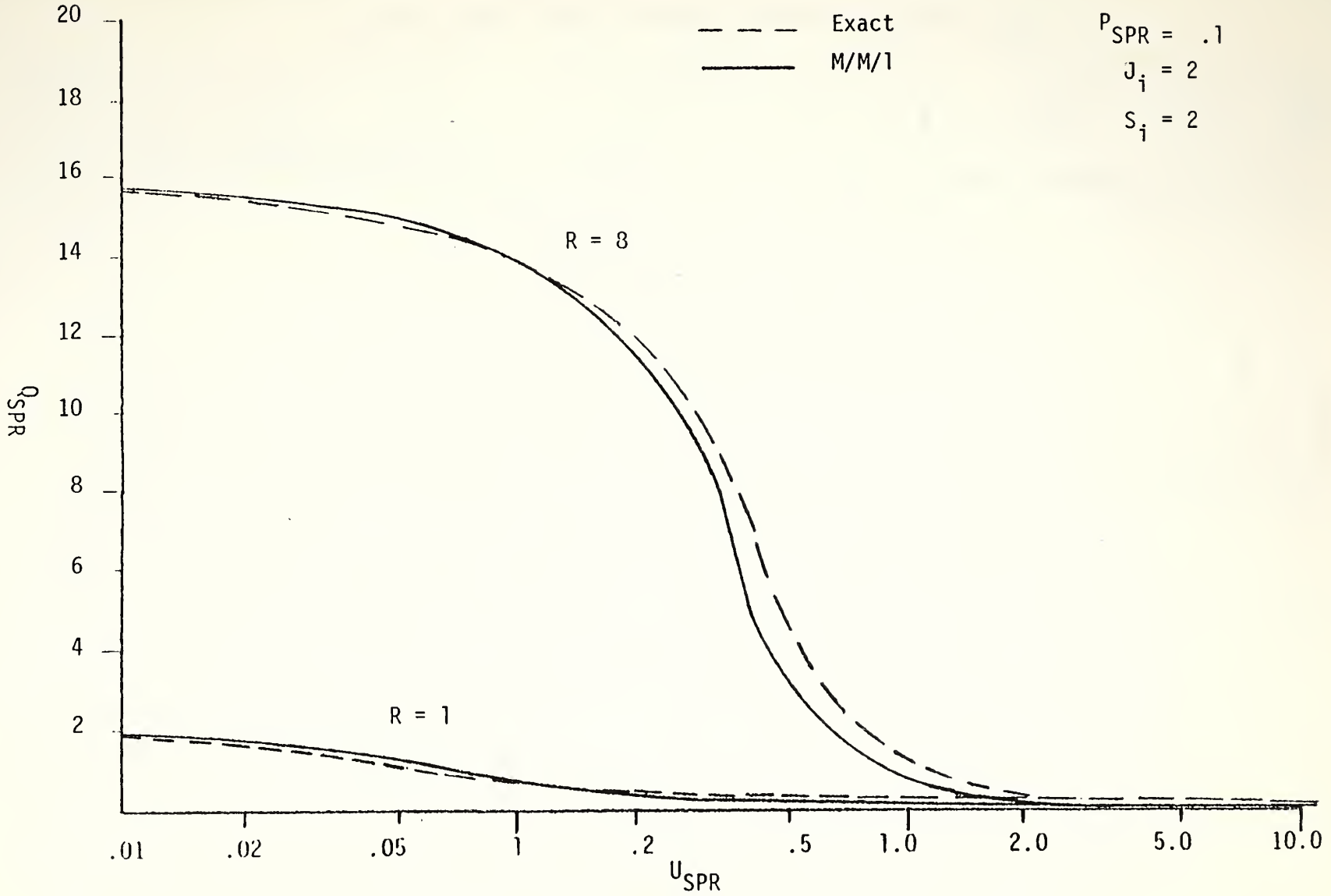

Figure IV-19. Exact and M/M/1 values for Q_{SPR} vs. U_{SPR}.

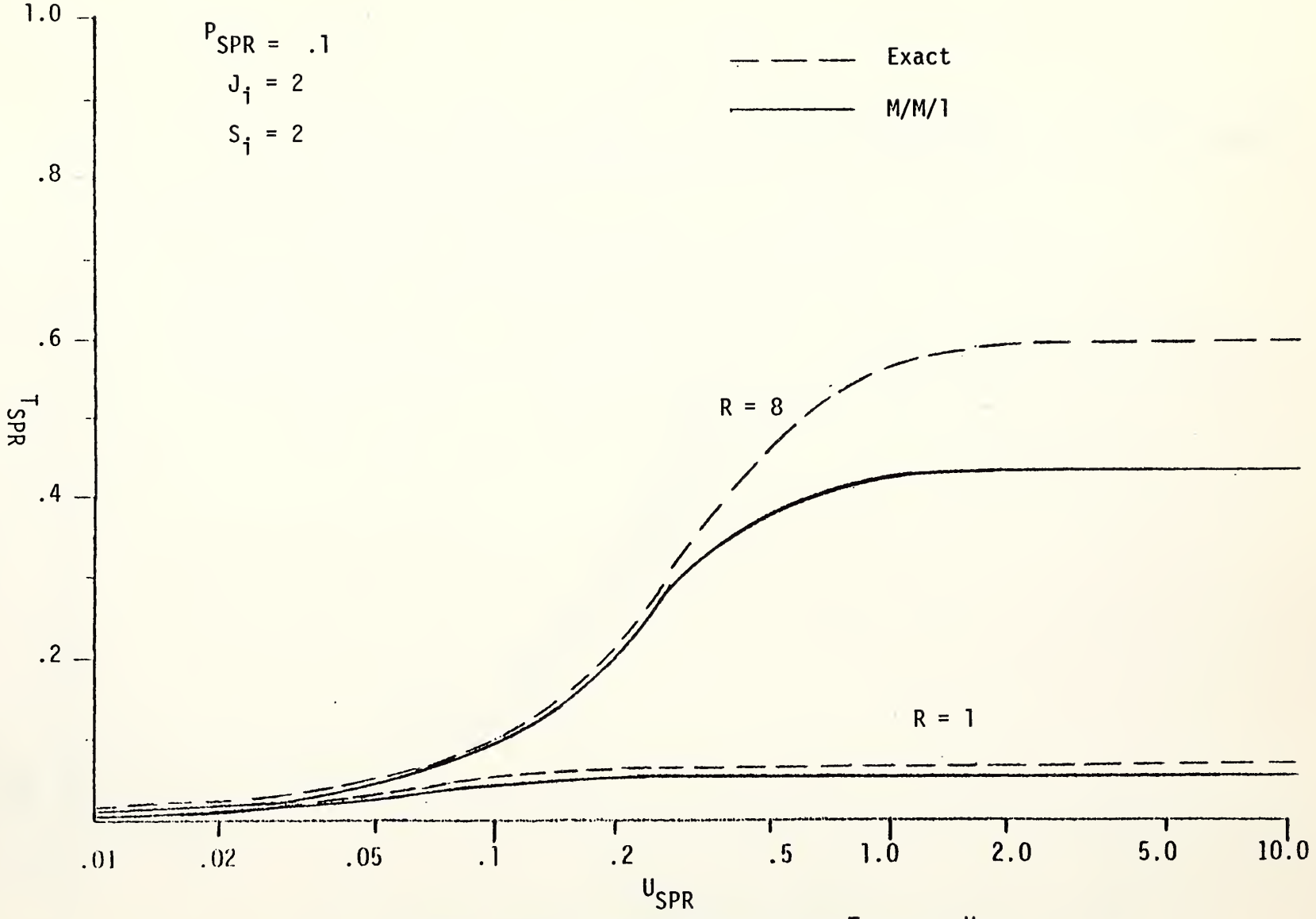

Figure IV-20. Exact and M/M/1 values for T_{SPR} vs. U_{SPR}.

From these relative error plots and statistics tables of the individual parameters, one can see that this approximation does possess some parameter sensitivity. The throughput performance measure does not indicate any sensitivity trends for the p_{SPR} and s parameters. For the J parameter a significant decreasing trend in mean and variance of the relative error is observed as J increases. No definite trend for the R parameter can readily be detected. Although the overall mean does decrease as R increases, the variance and the mean for those points not near device saturation do not. The p_{SPR} parameters are observed to behave similarly . As a result, the throughput performance measure exhibits some sensitivity to the J and R parameters.

In contrast to the throughput performance measure, the mean queue length has lower relative error values, approximately 12% for the CPU and 15% for the SPR. The mean queue lengths predicted by the approximate model underestimates those of the exact model for low values of ρ_{SPR} and overestimates them for high values of ρ_{SPR}. This accounts for the lower mean relative error values. In a manner similar to throughput, as saturation is approached the relative error becomes small. The variance of this relative error is higher than that for throughput, approximately $\pm 14\%$ vs. $\pm 6\%$. This can be seen from the scatter plots, figures IV-3 through IV-16, especially for mid-range values of ρ_{SPR}. This implies that the mean queue length performance measure is more sensitive to our approximation than is the throughput performance measure. This is consistent with results obtained by Buzen [BUZEN 77] in his use of single server approximations.

Observations of the limited relative error statistics from the tables and plots indivate some trends and sensitivities, although they are inconclusive. The mean queue length performance measure does indicate a decreasing sensitivity to the s parameter as this parameter increases. This may be credited to these PPUs handling a larger portion of the workload and, therefore, the CPU and SPR are less heavily loaded. Both devices do not indicate any sensitivity to the p_{SPR} parameter. The J parameter indicates a definite sensitivity trend in both mean and variance. As J increases the relative error decreases. Both devices indicate a trend for the u_{SPR} parameter, but in opposite directions. The CPU demonstrates a lower mean and variance as u_{SPR} increases, but then as u_{SPR} becomes relatively fast the mean and variance tend to reverse and increase. The SPR demonstrates a totally opposite response. The mean and variance increase with increasing u_{SPR} and then decrease. For the R parameter both devices demonstrate a trend, but again exhibit opposite reactions.

The CPU exhibits a decreasing mean as R increases, while the SPR trend is an increasing one. The variance of both devices do not indicate a clear trend for this parameter. Therefore, the mean queue length performance measure demonstrates a higher sensitivity to the J and R parameters than the throughput performance measure, and is also sensitive to the u_{SPR} parameter. In addition, the CPU and SPR demonstrate opposite sensitivity for the u_{SPR} and R parameters.

An error analysis of the approximation has been presented to aid designers and analysts when they apply this approximation. Although the error analysis is by no means elaborate or conclusive, some preliminary trends and sensitivities have been identified. This by far exceeds the error analysis presented to support other approximations in the literature. Further work to establish an accurate error function which incorporates all these parameters is still needed.

In conclusion, we feel that the approximate SCS model provides reasonable results with small computational requirements. The approximate model computation times are on the order of a few seconds vs. minutes, hours, or even days for the exact model. The throughput values computed by the approximate model are always less than or equal to the exact values and on the average 20% less, $\pm$6%. While the mean queue length underestimates the exact value for low ρ_{SPR} and overestimates it for high values of ρ_{SPR}, the average is approximately 12% to 15%, $\pm$14% , with the greatest variation occuring in the mid-range of ρ_{SPR}.

V. ANALYSIS OF MODULAR EXPANSION

A. Exact Analysis

In chapter III, a queueing network model was developed for an architecture consisting of independent computing systems (ISCs) sharing a single device. In chapter IV, a much less complex approximate model for this type of architecture was introduced. Of interest for this architecture is the effect when the system is incrementally expanded by the addition of ICSs. Expansion of this type places a heavier load on the shared device, causing degraded service to each ICS. This introduces a dual problem. First, for a given configuration and a specific expansion, what is the degradation in service that results? This can be determined by using either of the models to compute any of the previously discussed performance measures for both the before and after cases. By comparing these measures against each other, as well as the requirements of the facility, one may determine if the degradation is significant and acceptable.

The second problem occurs when it is determined that the degradation is not acceptable. The alternatives then are to either forego the expansion or augment the shared device to increase its processing rate. The problem is then one of determining the amount by which the processing rate of the shared device must be increased to maintain the current level of service being delivered to each of the ICSs.

Both the exact and approximate models are used to develop corresponding relationships between adding ICSs and increasing the processing rate of the shared device. A before and after comparison of a response performance measure for a modular expansion of a balanced system is considered. The performance measure of interest here is the mean cycle time. This is the mean time of a renewal interval. This interval begins when a job enters the CPU queue, and terminates when that same job next enters the CPU queue again. This is a measure of the average time spent at each device (both waiting for and being processed), weighted by the probability of visiting that device. Each job, in general, will require many different cycles to complete its processing task, of concern here is the mean time for this performance measure.

From Little's result (W = Q/a), the mean wait time (W) spent at a device (in queue and processing) can be determined if the mean queue length (Q) and the mean arrival rate (a) of that device are known. The mean queue lengths for the exact SCS model, from (40) and (41) of chapter III, are

$$(1) \qquad Q_{i,j} = \frac{1}{G(J)} \sum_{k=1}^{J_i} x_{i,j}^{k} G(J-kd_i) \quad , \text{ for } i,j \neq 0 \text{ , and}$$

$$(2) \qquad Q_0 = \frac{1}{G(J)} \sum_{k=1}^{K} k! \, k \left\{ \sum_{\substack{R \\ \Sigma n_{i,0}=k \\ i=1}} \prod_{r=1}^{R} h(r;J_r-n_{r,0}) \right\}.$$

The departure rate of a server whose service process is exponential is equal to its arrival rate [BURKE 56, FINCH 59,BURKE 72, MUNTZ 73, KLIENR 76]. As noted before, the throughput performance measure is actually the device departure rate, and from (43) and (44) of chapter III is

$$(3) \qquad T_{i,j} = u_{i,j} A_{i,j} = \frac{G(J-d_i)}{G(J)} e_{i,j} \quad , \quad \text{for } i,j \neq 0 \quad , \text{ and}$$

$$(4) \qquad T_0 = u_0 A_0 = u_0 \left\{ 1 - \frac{\prod_{i=1}^{R} g_i(J_i)}{G(J)} \right\} \quad .$$

Therefore, an expression for the wait time at a device is

$$(5) \qquad W_{i,j} = Q_{i,j} / T_{i,j} \quad , \text{ for } i > 0 \text{ and } j \geq 0 \quad .$$

From this an expression can be formulated for the mean cycle time of a job assigned to an ICS as

$$W_i = \sum_{j=0}^{s_i} p_{i,j}\ W_{i,j} \qquad \text{, for } i>0 \text{ and } j \geq 0$$

(6)

$$= \sum_{j=0}^{s_i} p_{i,j}\ Q_{i,j}\ /T_{i,j}$$

$$= p_{i,0}\ Q_0 /T_0 \ + p_{i,1}\ Q_{i,1}\ /T_{i,1} \ + \sum_{j=2}^{s_i} p_{i,j}\ Q_{i,j}\ /T_{i,j}$$

$$= p_{i,SPR}\ Q_{SPR} /T_{SPR} \ + p_{i,CPU}\ Q_{i,CPU}\ /T_{i,CPU} \ + \sum_{j=2}^{s_i} p_{i,j}\ Q_{i,j}\ /T_{i,j}$$

Given an SCS system consisting of R ICSs, the desire is at least to maintain the same mean cycle time after a modular expansion resulting in an SCS system of R' ICSs. It is assumed that this can be accomplished by increasing the mean processing rate of the SPR, and further that this increase can be expressed as some multiplicative factor β. This relation can be expressed using the mean cycle time performance measure as

$$W_i(\beta\ u_{SPR}, R') \ \leq \ W_i(\ u_{SPR}, R) \quad , \qquad \text{or}$$

(7)

$$p_{i,SPR}\ Q_{SPR}(\beta\ u_{SPR}, R')\ /T_{SPR}(\beta\ u_{SPR}, R') \ + p_{i,CPU}\ Q_{i,CPU}\ /T_{i,CPU} \ + \sum_{j=2}^{s_i} p_{i,j}\ Q_{i,j}\ /T_{i,j}$$

$$\leq \ p_{i,SPR}\ Q_{SPR}\ (\ u_{SPR}, R)/T_{SPR}(\ u_{SPR}, R) \ + p_{i,CPU}\ Q_{i,CPU}\ /T_{i,CPU} \ + \sum_{j=2}^{s_i} p_{i,j}\ Q_{i,j}\ /T_{i,j}$$

where

$$R' \ > \ R \ \geq \ 1 \qquad \text{, and}$$

(8a) $$\beta \ = \ \alpha\ R'/R \qquad \text{, or}$$

(8b) $$\beta = 1 + \alpha\ (R'/R - 1) \quad .$$

By successfully increasing the SPR processing rate to handle the incremental load of R'-R additional ICSs, it can be assumed that the wait at each device within each ICS remains the same. This results in (7) reducing to the wait at the SPR only, which is

(9)
$$p_{i,SPR}\, Q_{SPR}(\beta\, u_{SPR}, R')\,/T_{SPR}(\beta\, u_{SPR}, R') \leq p_{i,SPR}\, Q_{SPR}(u_{SPR}, R)/T_{SPR}(u_{SPR}, R)$$
$$Q_{SPR}(\beta\, u_{SPR}, R')\,/T_{SPR}(\beta\, u_{SPR}, R') \leq Q_{SPR}(u_{SPR}, R)/T_{SPR}(u_{SPR}, R)$$

By reformulating (4) and substituting equation (31) of Chapter III we obtain

$$T_{SPR} = u_{SPR}\, A_{SPR} = u_{SPR} \left\{ 1 - \frac{\prod_{i=1}^{R} g_i(J_i)}{G(J)} \right\} \quad .$$

(10)
$$= u_{SPR}\, \frac{1}{G(J)} \sum_{k=1}^{K} k! \left\{ \sum_{\sum_{i=1}^{R} n_{i,0}=k} \prod_{r=1}^{R} h(r; J_r - n_{r,0}) \right\} \quad .$$

Substituting (2) and (10) into (9) a complete expression for the inequality is obtained of the form

$$\frac{\frac{1}{G(J')} \sum_{k'=1}^{K'} k'!\, k' \left\{ \sum_{\substack{R' \\ \sum n_{i,0}=k' \\ i=1}} \prod_{r=1}^{R'} h(r;J_r-n_{r,0}) \right\}}{\beta\, u_{SPR} \frac{1}{G(J')} \sum_{k'=1}^{K'} k'! \left\{ \sum_{\substack{R' \\ \sum n_{i,0}=k' \\ i=1}} \prod_{r=1}^{R'} h(r;J_r-n_{r,0}) \right\}}$$

$$\leq \frac{\frac{1}{G(J)} \sum_{k=1}^{K} k!\, k \left\{ \sum_{\substack{R \\ \sum n_{i,0}=k \\ i=1}} \prod_{r=1}^{R} h(r;J_r-n_{r,0}) \right\}}{u_{SPR} \frac{1}{G(J)} \sum_{k=1}^{K} k! \left\{ \sum_{\substack{R \\ \sum n_{i,0}=k \\ i=1}} \prod_{r=1}^{R} h(r;J_r-n_{r,0}) \right\}}$$

Expanding the h functions in the above expression, from their definition in chapter III.B between equations (14) and (15), results in

$$\frac{\frac{1}{G(J')} \sum_{k'=1}^{K'} k'!\, k' \left\{ \sum_{\substack{R' \\ \sum n_{i,0}=k' \\ i=1}} \prod_{r=1}^{R'} \frac{x'_{r,0}{}^{n_{r,0}}}{n_{r,0}!} g_r(J_r-n_{r,0}) \right\}}{\beta\, u_{SPR} \frac{1}{G(J')} \sum_{k'=1}^{K'} k'! \left\{ \sum_{\substack{R' \\ \sum n_{i,0}=k' \\ i=1}} \prod_{r=1}^{R'} \frac{x'_{r,0}{}^{n_{r,0}}}{n_{r,0}!} g_r(J_r-n_{r,0}) \right\}} \leq$$

$$\frac{\frac{1}{G(J)} \sum_{k=1}^{K} k!\, k \left\{ \sum_{\substack{R \\ \sum_{i=1} n_{i,0}=k}} \prod_{r=1}^{R} \frac{x_{r,0}^{\,n_{r,0}}}{n_{r,0}!}\, g_r(J_r - n_{r,0}) \right\}}{u_{SPR}\, \frac{1}{G(J)} \sum_{k=1}^{K} k! \left\{ \sum_{\substack{R \\ \sum_{i=1} n_{i,0}=k}} \prod_{r=1}^{R} \frac{x_{r,0}^{\,n_{r,0}}}{n_{r,0}!}\, g_r(J_r - n_{r,0}) \right\}} \leq$$

where

$K' = R' J_i$,

$K = R J_i$,

$J' = (J_1, \ldots, J_{R'})$,

$J = (J_1, \ldots, J_R)$,

$J_i = J_j$, for all i and j , and

$x'_{r,0}{}^{n_{r,0}} = (x_{r,0}/\beta)^{n_{r,0}}$.

Cancelling similar terms, cross multiplying, and moving everything to the left hand side of the inequality results in

$$\frac{\sum_{k=1}^{K} k! \left\{ \sum_{\substack{R \\ \sum_{i=1} n_{i,0}=k}} \prod_{r=1}^{R} \frac{x_{r,0}^{\,n_{r,0}}}{n_{r,0}!}\, g_r(J_r - n_{r,0}) \right\}}{\beta \sum_{k'=1}^{K'} k'! \left\{ \sum_{\substack{R' \\ \sum_{i=1} n_{i,0}=k'}} \prod_{r=1}^{R'} \frac{x'_{r,0}{}^{n_{r,0}}}{n_{r,0}!}\, g_r(J_r - n_{r,0}) \right\}} \qquad (11)$$

$$(11) \qquad - \frac{\displaystyle\sum_{k=1}^{K} k!\, k \left\{ \sum_{\substack{\sum_{i=1}^{R} n_{i,0}=k}} \prod_{r=1}^{R} \frac{x_{r,0}^{\,n_{r,0}}}{n_{r,0}!}\, g_r(J_r-n_{r,0}) \right\}}{\displaystyle\sum_{k'=1}^{K'} k'!\, k' \left\{ \sum_{\substack{\sum_{i=1}^{R'} n_{i,0}=k'}} \prod_{r=1}^{R'} \frac{x'_{r,0}^{\,n_{r,0}}}{n_{r,0}!}\, g_r(J_r-n_{r,0}) \right\}} \leq 0$$

Inspecting (11) we notice that all terms may be moved into the innermost summation. Noting the similarity of the summation in both numerators and denomenators a simplifying notation is introduced. Let

$$b_k = \left\{ \sum_{\substack{\sum_{i=1}^{R} n_{i,0}=k}} k! \prod_{r=1}^{R} \frac{x_{r,0}^{\,n_{r,0}}}{n_{r,0}!}\, g_r(J_r-n_{r,0}) \right\} \quad , \text{ and}$$

$$b'_{k'} = \left\{ \sum_{\substack{\sum_{i=1}^{R'} n_{i,0}=k'}} k'! \prod_{r=1}^{R'} \frac{x'_{r,0}^{\,n_{r,0}}}{n_{r,0}!}\, g_r(J_r-n_{r,0}) \right\} \quad .$$

Substituting this notation into (11) yields

$$\frac{\sum_{k=1}^{K} b_k}{\beta \sum_{k'=1}^{K'} b'_{k'}} - \frac{\sum_{k=1}^{K} k\, b_k}{\sum_{k'=1}^{K'} k'\, b'_{k'}} \leq 0$$

Separating the common outer summation term results in

$$(12) \quad \sum_{k=1}^{K} b_k \left[\frac{1}{\beta \sum_{k'=1}^{K'} b'_{k'}} - \frac{k}{\sum_{k'=1}^{K'} k'\, b'_{k'}} \right] \leq 0$$

The b_k's of this inequality are always positive. Therefore, for the inequality to be satisfied requires the inner term to act as a weighting function and force the entire expression to be non-positive. Although a solution may exist for β which will satisfy the equality, no obvious method of obtaining it is apparent. In an attempt to satisfy the inequality and obtain a lower bound for β we will investigate the situation when the the inner term is always ≤ 0. By inspection of (12) we notice that within the inner term, except for k in the numerator, the other terms are independent of k. As a result this inner term achieves its maximum value at the minimum value of k, which is 1. If this term is ≤ 0 for its maximum value, it is ≤ 0 for all values of k, and the inequality is satisfied. This leaves the following relation for a lower bound solution for β:

$$\left[\frac{1}{\beta \sum_{k'=1}^{K'} b'_{k'}} - \frac{1}{\sum_{k'=1}^{K'} k'\, b'_{k'}} \right] \leq 0 \; .$$

By rearranging the above we obtain

$$\left[\sum_{k'=1}^{K'} k' \, b'_{k'} - \beta \sum_{k'=1}^{K'} b'_{k'} \right] \leq 0 \quad .$$

Repeating the previously applied separation process yields

$$\sum_{k'=1}^{K'} b'_{k'} \quad [\, k' - \beta \,] \leq 0 \quad .$$

Again applying our previous arguments we obtain a further lower bound solution to the expression, since b'_k is always positive. This inner term achieves its maximum value at the maximum value of k', which is $K' = R' J_i$. Therefore, the inequality is always satisfied if it is satisfied for k' = K'. This yields the following lower bound solution for β :

$$\beta = K' = R' J_i \quad .$$

Solving for α by substituting (8a) into the above yields $\alpha = R J_i$.

The interpretation of this result implies that by increasing the processing rate of the SPR by a factor commensurate with the total resulting number of jobs in the entire system one will be assured no degradation in response occurs as compared to the response prior to the expansion. Unfortunately this is such an extremely high lower bound that the result is not very useful.

In retrospect, based on the results derived in the next section, if one repeats this procedure and differs only by substituting in (12) the maximum (K = RJ) rather than its minimum (1) value of k, the result obtained is :

$$\beta = R'/R \quad \text{and from either (8a) or (8b)} \quad \alpha = 1 \, .$$

This result is much more intuitively appealing due to its linear one-to-one relation, but its derivation cannot be substantiated from the above equations. Although $b_k < b_{k+1}$ for all k with b_k increasing factorially, and the inner term of (12) is linearly increasing in the negative direction with k, this is not sufficient to conclude that

$$b_K \left[\frac{1}{\beta \sum_{k'=1}^{K'} b'_{k'}} - \frac{K}{\sum_{k'=1}^{K'} k' \, b'_{k'}} \right] \geq \sum_{k=1}^{K-1} b_k \left[\frac{1}{\beta \sum_{k'=1}^{K'} b'_{k'}} - \frac{k}{\sum_{k'=1}^{K'} k' \, b'_{k'}} \right] .$$

B. Approximate Analysis

Analysis of the expression for mean cycle time of the exact SCS model has resulted in a dissappointingly high lower bound for β. In this section the approximate SCS model developed in chapter IV is used to perform a similar analysis.

We can immediately write a similar expression for (6), from (9) and (11) of chapter IV, as

$$W_i = e_{SPR_i} W_{SPR} + W_{CPU_i} + \sum_{j=2}^{s_i} e_{PPU_{i,j}} W_{PPU_{i,j}}$$

$$(13) \qquad = \frac{e_{SPR_i} Q_{SPR}}{T_{SPR}} + \frac{Q_{CPU_i}}{a_{CPU_i}} + \sum_{j=2}^{s_i} \frac{e_{PPU_{i,j}} Q_{PPU_{i,j}}}{T_{PPU_{i,j}}}$$

$$= \frac{1}{a_{CPU_i}} \left\{ Q_{SPR}/R + Q_{CPU_i} + \sum_{j=2}^{s_i} Q_{PPU_{i,j}} \right\}$$

The intent is to maintain the mean cycle time of a job after a modular expansion. Therefore, a relation similar to (7) can be established as

$$W_i(\beta\, u_{SPR}, R') \leq W_i(u_{SPR}, R) \quad , \quad \text{or}$$

(14)

$$\frac{1}{a'_{CPU_i}} \left\{ Q'_{SPR}/R' + Q_{CPU_i} + \sum_{j=2}^{s_i} Q_{PPU_{i,j}} \right\} \leq \frac{1}{a_{CPU_i}} \left\{ Q_{SPR}/R + Q_{CPU_i} + \sum_{j=2}^{s_i} Q_{PPU_{i,j}} \right\}$$

Assuming that the processing rate of the SPR is successfully increased to satisfy the above inequality, it can be assumed that the wait at each device within each ICS remains the same. This further implies that a_{CPU_i} also remains the same. This results in reducing (14) to

$$\frac{1}{a'_{CPU_i}} \left\{ Q'_{SPR}/R' \right\} \leq \frac{1}{a_{CPU_i}} \left\{ Q_{SPR}/R \right\}$$

(15)

$$Q'_{SPR}/R' \leq Q_{SPR}/R$$

$$\frac{1/R'}{(\beta\, u_{SPR}/R'\, p_{SPR_i}\, a_{CPU_i} - 1)} \leq \frac{1/R}{(u_{SPR}/R\, p_{SPR_i}\, a_{CPU_i} - 1)}$$

By inspection of (15) one can determine that by maintaining a constant ratio of

$$\beta\, u_{SPR}/R' = C \quad , \text{ where } C = u_{SPR}/R \quad ,$$

results in

$$\beta = R'/R \quad \text{and from (8a)} \quad \alpha = 1 \quad ,$$

which satisfies the inequality and actually improves the cycle time rather than just maintaining it. This was conjectured, but not proven, at the end of the the preceeding section. Maintaining the assumption that a_{CPU_i} remains the same, the equality of (15) is solved for the required β.

$$R\,(\,u_{SPR}/R\;p_{SPR_i}\;a_{CPU_i}\;-\;1)\;\;=\;\;R'\,(\beta\;u_{SPR}/R'\;p_{SPR_i}\;a_{CPU_i}\;-\;1)$$

$$(\beta\;-\;1)\;\;=\;\;(p_{SPR_i}\;a_{CPU_i}\;/u_{SPR})\;\;(R'\;-\;R)$$

Noting that $\rho_{SPR} = R\;p_{SPR_i}\;a_{CPU_i}\;/u_{SPR}$ and subtituting it into the above yields

$$(\beta\;-\;1)\;\;=\;\;(\rho_{SPR}/R)\;(R'\;-\;R)\;\;.$$

This may also be expressed as

$$\beta\;\;=\;1\;+\;(\rho_{SPR}/R)\;(R'\;-\;R)$$

(16)

$$=\;1\;+\;\rho_{SPR}\;(R'/R\;\;-\;1)$$

From this, and (8b), $\alpha\;\;=\rho_{SPR}$ is obtained.

To evaluate the accuracy of the wait time performance measures and, therefore, the accuracy of this modular expansion result, we present an error analysis similar to that of chapter IV. The relative error scatter and mean value plots of the CPU and SPR mean wait time are presented in Figures V-1 and V-2 for the entire sample space. The relative error statistics for this performance measure are listed by parameter in Tables V-1 and V-2. Representative comparison curves for the exact and approximate models are plotted in Figures V-3 and V-4.

As can be seen from Tables V-1 and V-2, this performance measure exhibits some parameter sensitivity. Some sensitivity is indicated for the s parameter. The CPU indicates a decreasing error as s increases, while the SPR show an opposite response. This sensitivity is attributed to the large number of PPUs spawned by increasing s, resulting in a greater portion of the workload begin handled by the PPUs, decreasing the SPR and CPU workload.

Both devices demonstrate a sensitivity to the p_{SPR} parameter. The CPU does not exhibit any trend, while the SPR shows an increasing error trend with increasing p_{SPR}. Both devices exhibit a decreasing error as the J parameter increases. Both devices demonstrate an opposite sensitivity trend to the u_{SPR} and the R parameters. The CPU shows an increasing mean error,

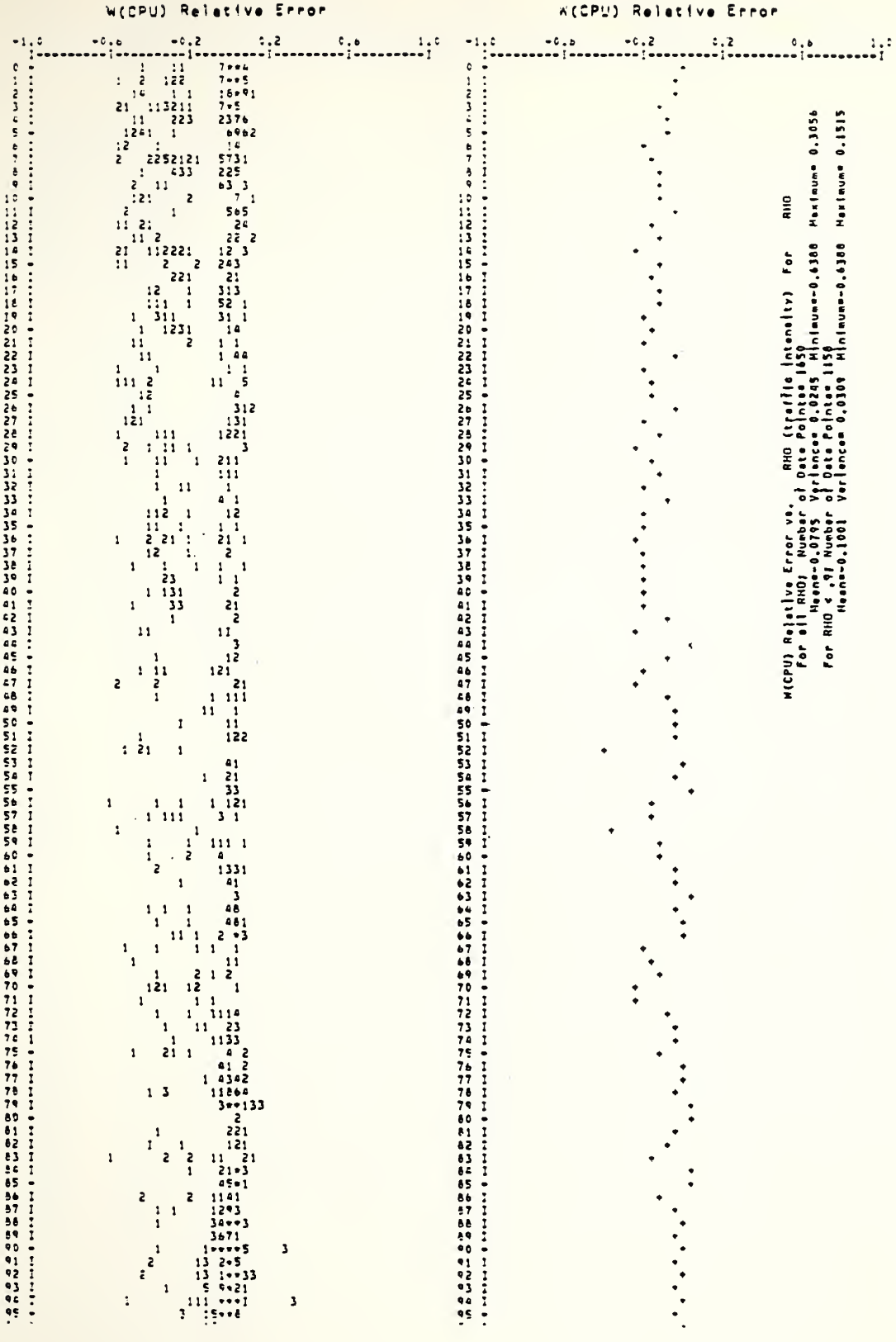

Figure V-1. Relative error plot of W_{CPU} for all points

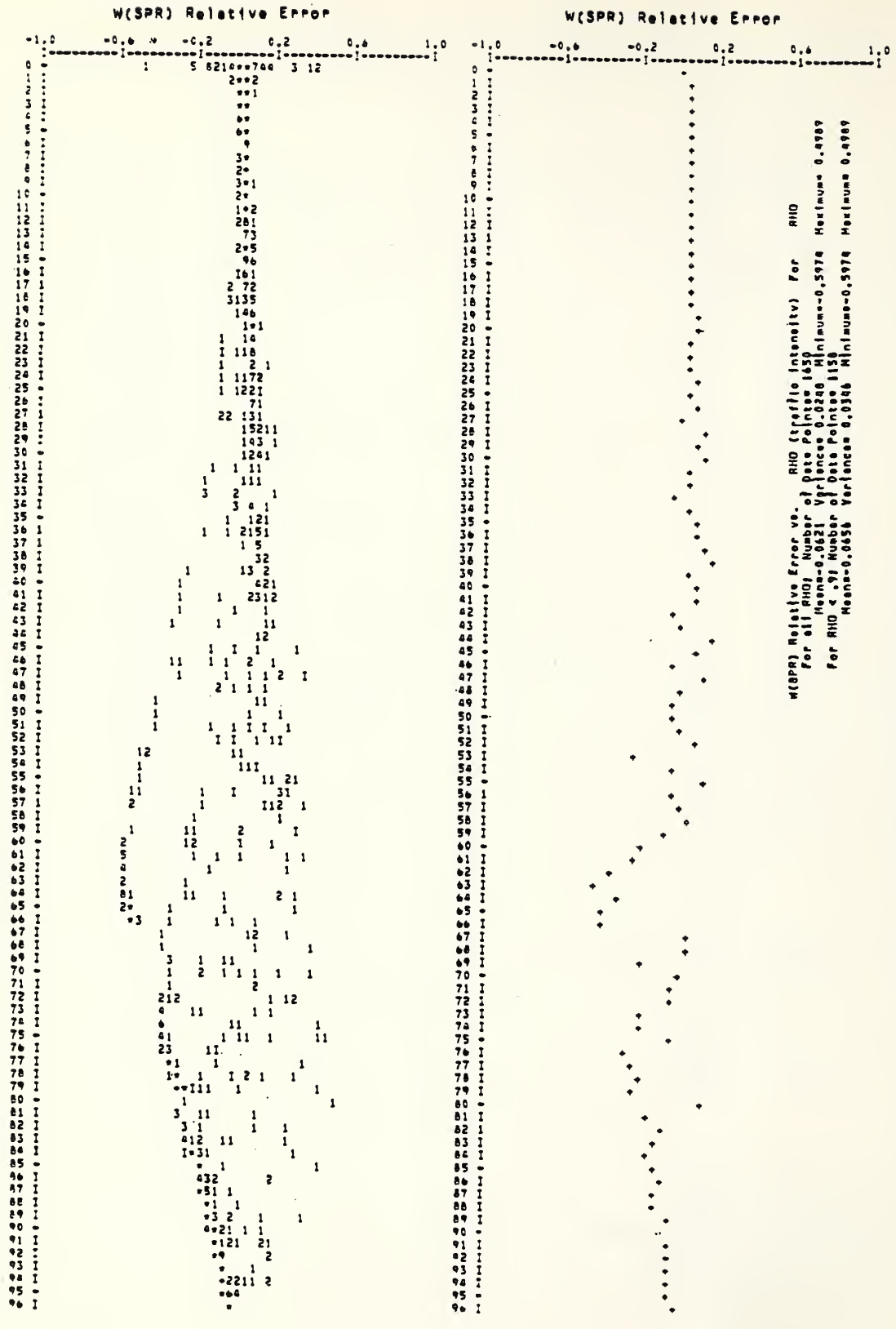

Figure V-2. Relative error plot of W_{SPR} for all points

ρ	Points	Mean	Variance	Minimum	Maximum
—	1650/1158	-0.0795/-0.1001	0.0245/ 0.0309	-0.6388/-0.6388	0.3056/ 0.1515

p_{SPR}	Points	Mean	Variance	Minimum	Maximum
0.10	330/ 266	-0.0739/-0.0857	0.0203/ 0.0238	-0.4489/-0.4489	0.0798/ 0.0798
0.25	330/ 260	-0.0686/-0.0778	0.0247/ 0.0297	-0.5893/-0.5893	0.0787/ 0.0787
0.50	330/ 208	-0.1526/-0.2246	0.0378/ 0.0390	-0.5647/-0.5647	0.3056/ 0.0777
0.75	330/ 227	-0.0000/ 0.0075	0.0026/ 0.0021	-0.1042/-0.0947	0.2500/ 0.0970
0.90	330/ 197	-0.1025/-0.1416	0.0253/ 0.0352	-0.6388/-0.6388	0.1515/ 0.1515

J	Points	Mean	Variance	Minimum	Maximum
2.00	600/ 458	-0.1169/-0.1447	0.0335/ 0.0387	-0.6388/-0.6388	0.3056/ 0.1515
4.00	600/ 411	-0.0682/-0.0818	0.0206/ 0.0264	-0.5352/-0.5074	0.3056/ 0.0685
6.00	450/ 289	-0.0448/-0.0555	0.0147/ 0.0196	-0.4723/-0.3813	0.0956/ 0.0956

s	Points	Mean	Variance	Minimum	Maximum
2.00	550/ 387	-0.1000/-0.1275	0.0287/ 0.0353	-0.6388/-0.6388	0.3056/ 0.1515
6.00	550/ 389	-0.0420/-0.0492	0.0138/ 0.0175	-0.4914/-0.4914	0.3056/ 0.1515
11.00	550/ 382	-0.0966/-0.1243	0.0291/ 0.0365	-0.5893/-0.5893	0.3056/ 0.1515

R	Points	Mean	Variance	Minimum	Maximum
1.00	450/ 450	-0.0786/-0.0786	0.0271/ 0.0271	-0.5814/-0.5814	0.0956/ 0.0956
2.00	450/ 383	-0.0787/-0.0927	0.0248/ 0.0277	-0.5811/-0.5811	0.1515/ 0.1515
5.00	450/ 213	-0.0799/-0.1287	0.0215/ 0.0354	-0.6239/-0.6239	0.2500/ 0.0816
8.00	300/ 112	-0.0815/-0.1575	0.0250/ 0.0431	-0.6388/-0.6388	0.3056/ 0.0670

u_{SPR}	Points	Mean	Variance	Minimum	Maximum
0.01	165/ 76	0.0089/ 0.0133	0.0069/ 0.0017	-0.1111/-0.0622	0.3056/ 0.1515
0.02	165/ 79	-0.0152/-0.0008	0.0011/ 0.0005	-0.1042/-0.0617	0.0426/ 0.0402
0.05	165/ 85	-0.0084/-0.0003	0.0007/ 0.0006	-0.1044/-0.1044	0.0798/ 0.0798
0.10	165/ 92	-0.0123/-0.0106	0.0018/ 0.0029	-0.2889/-0.2889	0.0787/ 0.0787
0.20	165/ 100	-0.0257/-0.0292	0.0056/ 0.0087	-0.4051/-0.4051	0.0777/ 0.0777
0.50	165/ 120	-0.0745/-0.0856	0.0159/ 0.0203	-0.5105/-0.5105	0.0956/ 0.0956
1.00	165/ 131	-0.1288/-0.1389	0.0272/ 0.0320	-0.5644/-0.5644	0.0771/ 0.0771
2.00	165/ 147	-0.1721/-0.1691	0.0390/ 0.0422	-0.5893/-0.5893	0.0805/ 0.0805
5.00	165/ 163	-0.1876/-0.1840	0.0466/ 0.0461	-0.6388/-0.6388	0.0828/ 0.0828
10.00	165/ 165	-0.1797/-0.1797	0.0445/ 0.0445	-0.5898/-0.5898	0.0819/ 0.0819

Table V-1. Relative Error statistics for W_{CPU}, for both all $\rho/\rho < .90$.

ρ	Points	Mean	Variance	Minimum	Maximum
—	1650/1158	-0.0621/-0.0656	0.0248/ 0.0346	-0.5974/-0.5974	0.4989/ 0.4989

p_{SPR}	Points	Mean	Variance	Minimum	Maximum
0.10	330/ 266	-0.0367/-0.0325	0.0188/ 0.0227	-0.5974/-0.5974	0.3547/ 0.3547
0.25	330/ 260	-0.0299/-0.0247	0.0241/ 0.0298	-0.5810/-0.5810	0.4402/ 0.4402
0.50	330/ 208	-0.0899/-0.1105	0.0239/ 0.0358	-0.5930/-0.5930	0.3320/ 0.3320
0.75	330/ 227	-0.0604/-0.0634	0.0266/ 0.0382	-0.5774/-0.5774	0.4989/ 0.4989
0.90	330/ 197	-0.0936/-0.1194	0.0275/ 0.0436	-0.5916/-0.5916	0.4262/ 0.4262

J	Points	Mean	Variance	Minimum	Maximum
2.00	600/ 458	-0.1116/-0.1206	0.0370/ 0.0480	-0.5974/-0.5974	0.3547/ 0.3547
4.00	600/ 411	-0.0367/-0.0378	0.0187/ 0.0267	-0.3979/-0.3979	0.4989/ 0.4989
6.00	450/ 289	-0.0299/-0.0178	0.0118/ 0.0166	-0.3018/-0.3018	0.4262/ 0.4262

s	Points	Mean	Variance	Minimum	Maximum
2.00	550/ 387	-0.0518/-0.0515	0.0265/ 0.0368	-0.5811/-0.5811	0.4262/ 0.4262
6.00	550/ 389	-0.0651/-0.0688	0.0235/ 0.0327	-0.5930/-0.5930	0.4072/ 0.4072
11.00	550/ 382	-0.0693/-0.0766	0.0244/ 0.0342	-0.5974/-0.5974	0.4989/ 0.4989

R	Points	Mean	Variance	Minimum	Maximum
1.00	450/ 450	-0.1553/-0.1553	0.0401/ 0.0401	-0.5974/-0.5974	0.3714/ 0.3714
2.00	450/ 383	-0.0767/-0.0738	0.0172/ 0.0201	-0.3964/-0.3964	0.3759/ 0.3759
5.00	450/ 213	-0.0055/ 0.0462	0.0100/ 0.0142	-0.2500/-0.2500	0.4262/ 0.4262
8.00	300/ 112	0.0147/ 0.1103	0.0117/ 0.0140	-0.0946/-0.0730	0.4989/ 0.4989

u_{SPR}	Points	Mean	Variance	Minimum	Maximum
0.01	165/ 76	-0.1575/-0.2698	0.0208/ 0.0209	-0.5974/-0.5974	-0.0308/-0.0549
0.02	165/ 79	-0.1517/-0.2512	0.0223/ 0.0268	-0.5811/-0.5811	0.1923/ 0.1923
0.05	165/ 85	-0.1346/-0.2062	0.0283/ 0.0432	-0.5797/-0.5797	0.4110/ 0.4110
0.10	165/ 92	-0.1225/-0.1717	0.0278/ 0.0438	-0.5930/-0.5930	0.3184/ 0.3184
0.20	165/ 100	-0.1049/-0.1359	0.0279/ 0.0428	-0.5916/-0.5916	0.3126/ 0.3126
0.50	165/ 120	-0.0590/-0.0623	0.0215/ 0.0288	-0.5223/-0.5223	0.4072/ 0.4072
1.00	165/ 131	-0.0067/ 0.0001	0.0115/ 0.0137	-0.3395/-0.3395	0.4989/ 0.4989
2.00	165/ 147	0.0395/ 0.0443	0.0092/ 0.0094	-0.2571/-0.2571	0.4402/ 0.4402
5.00	165/ 163	0.0567/ 0.0553	0.0083/ 0.0083	-0.1921/-0.1921	0.4262/ 0.4262
10.00	165/ 165	0.0197/ 0.0197	0.0097/ 0.0097	-0.5172/-0.5172	0.3759/ 0.3759

Table V-2. Relative Error statistics for W_{SPR}, for both all $\rho/\rho < .90$.

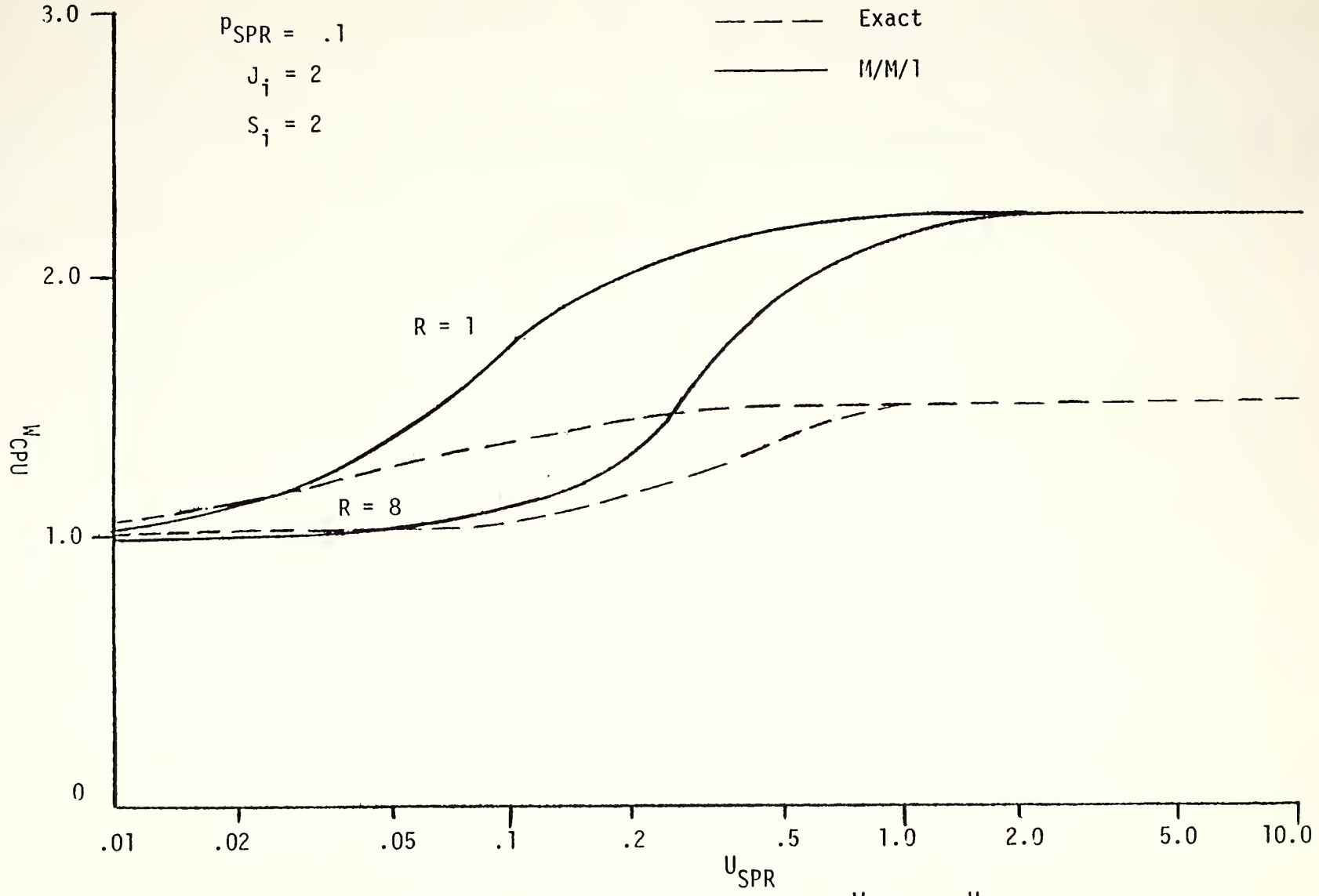

Figure V-3. Exact vs. M/M/1 plot of W_{CPU} vs. U_{SPR}.

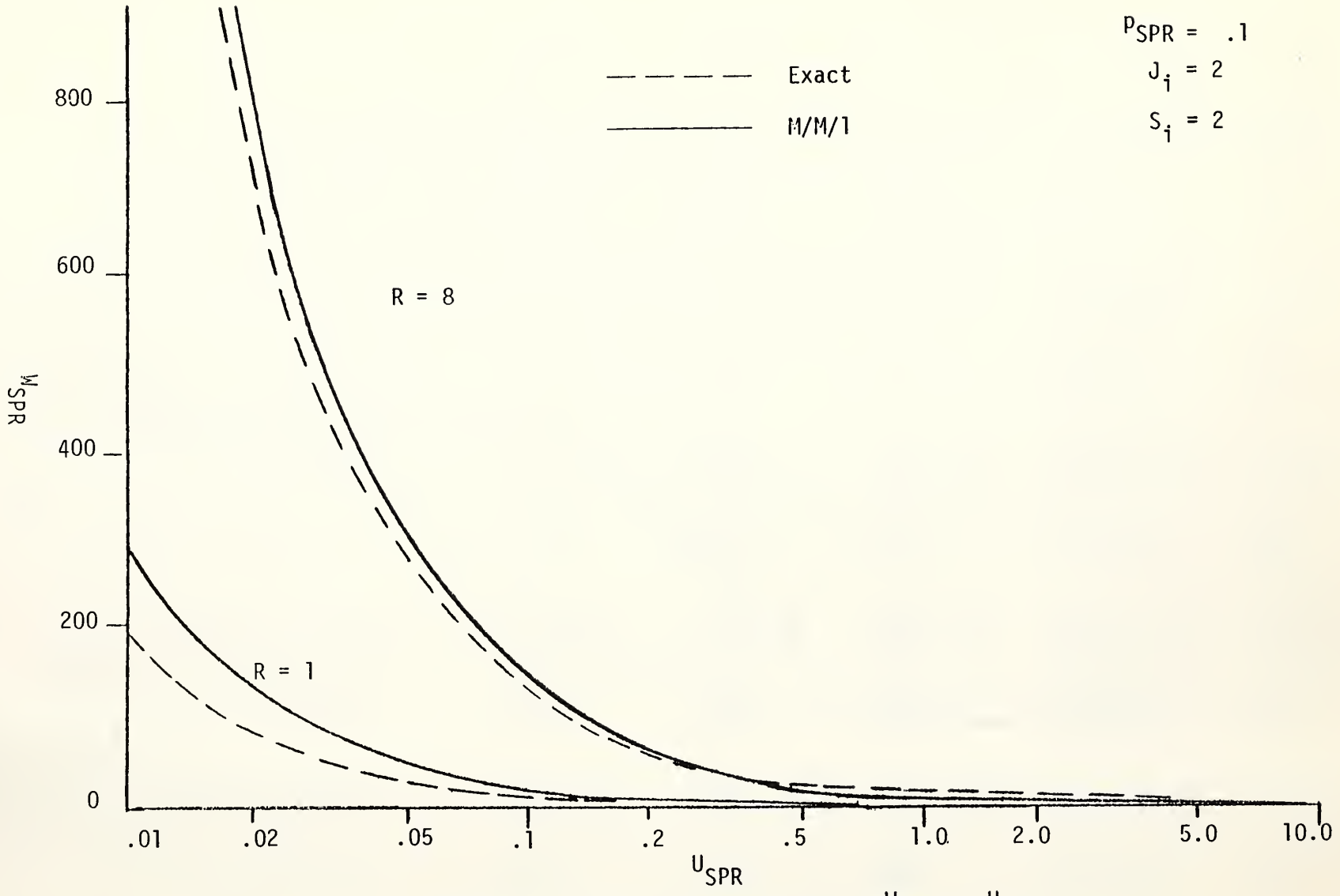

Figure V-4. Exact vs. M/M/1 plot of W_{SPR} vs. U_{SPR}.

while the SPR shows a decreasing error for increasing R. For increasing u_{SPR}, the CPU shows an increasing error, while the SPR first shows a decreasing error and then as u_{SPR} gets large, it tends to increase. Therefore, the wait time performance measure exhibits the greatest sensitivity of all the performance measures. Although wait time possesses the lowest mean relative error, it also possesses the highest variance, and demonstrates sensitivity to all parameters. This is reasonable to expect since the wait time is a function of the other two performance measures and, therefore, compounds their errors. In this case the variance was amplified while the mean was attenuated.

The interpretation of (16) implies that by increasing the processing rate of the SPR by ρ_{SPR}/R for each additional ICS in the resultant expanded system, the mean response time would be preserved when compared to that prior to the expansion. It is noted that $\rho_{SPR} < 1$ and, therefore, $\rho_{SPR}/R < 1$. This means that the incremental increase in SPR processing rate (per additional ICS), ρ_{SPR}/R, is only a fractional increase. This is a significant improvement over the upper bound derived in the previous section. In fact this is an improvement over the incremental increase of unity conjectured in the preceding section.

The unity increase in processing rate is quite intuitively appealing. For each additional ICS added, the SPR processing rate should increase by $1/R$ from the current processing rate. The newly derived incemental increase of ρ_{SPR}/R seems intuitively correct. Condider the fact ρ_{SPR} represents the current total traffic intensity to the SPR, ρ_{SPR}/R represents the portion of that traffic intensity generated by an ICS. Utilizing this perspective of traffic intensity, this fractional increase seems more credible. Since each ICS generates traffic proportional to ρ_{SPR}/R, each additional ICS would genarate additional traffic also proportional to ρ_{SPR}/R. Therefore, if the processing rate of the SPR is increased by that fractional increase in traffic, the response time should not increase.

There is an additional implication to this fractional increase in SPR processing rate, proportional to the increase in ICSs. A system of this class may undergo one or more modular expansions, encompassing a large increase in the number of ICSs. The extent of expansion will in general be limited technically by the maximum attainable processing rate of the SPR, which is dependent, for the most part, on the existing technology for that type of device. The processing

rate of applicable devices will generally span several orders of magnitude and may include a number of different technologies. Therefore, depending on a fractional processing rate increase, rather than a unity or greater increase, will allow a given technology to support a larger range of expansion. This minimizes, or at least delays, the implementaion and investment risk of changing device technology in a given system. Additionally it will also allow for a far greater maximum expansion range, since the overall existing technology limit will be approached at a much slower rate.

C. Applications

Two examples of the SPR architecture are presented to illustrate the utility of the approximate SCS model. The first example is a complex of multiple minicomputers linked to a common shared secondary memory subsystem by a local area network (LAN). This is applicable to engineering and scientific environments. The second example is a point-of-sales (POS) application. This is applicable to grocery and department stores, and has a direct analogy to certain office automation environments.

Example 1

Suppose The current processing system of a technical organization consists of 2 minicomputers (ICSs); each has an average multiprogramming level of eight, and both share a common secondary storage subsystem. The minicomputers are identical and each has a CPU and the same complement of four peripheral devices (PPU). These PPUs consist of (1) an input card reader (CR), (2) an output line printer (LP), (3) a private local disk (disk), and (4) a set of interactive devices (TTY).

Operationally, each ICS functions independently, processing both batch and interactive jobs. The composition of both types of jobs is the same, so no distinction between them is necessary. The set of interactive devices, as a set, has been characterized as a single device with an exponential service time distribution. Therefore, we may aggregate these interactive devices and represent them as a single device in our model. The local disk has two functions, (1) The system software resides there, and (2) during processing of a job it acts as a cache between the ICS and the SPR.

The jobs exhibits an exponential service time distribution at each device. The CPU mean service time of a job is 25 msec, which in the SCS model is normalized to 1. For each device the mean service time, its corresponding normalized value, and its estimated transition probabilities are as follows:

mean service time	normalized service time	estimated transition propabilities
u_{SPR} = 100 msec	u_0 = .25	p_0 = .25
u_{CPU} = 25 msec	u_1 = 1.0	p_1 = .05
u_{disk} = 25 msec	u_2 = 1.0	p_2 = .45
u_{TTY} = 2.5 sec	u_3 = .01	p_3 = .05
u_{LP} = 250 msec	u_4 = .10	p_4 = .10
u_{CR} = 250 msec	u_5 = .10	p_5 = .05

The installation is about to be modified. The organization is expanding and has determined a requirement to expand the processing complex by eightfold. They have decided to implement a local area network (LAN) which will allow interactive device access to the central processing complex from the desk of each employee. The bandwidth of the LAN is sufficiently high so that it will not be a bottleneck or cause any significant delay, therefore, the LAN may be neglected in our analysis. Due to existing software investment and staff familarity, the organization plans to retain the existing two minicomputers and modularly expand by adding identical ones as required.

The shared common secondary storage subsystem has been very successful for sharing, rather than duplicating, programs and data bases as well as providing an effective electronic mail system. Therefore, the retention and expansion of this facility is also planned. The organization is

expected to grow over the next two years and the modular expansion of the processing facilities is planned to coincide. The maximum planned processing expansion is for a complex of 16 minicomputers. Two minicomputers are currently being acquired to bring the complex to 4 minicomputers.

The current SPR does not have sufficient speed or capacity to handle the planned expansion. It is desired to size the SPR so that its current mean response time is maintained. In addition, there is potential for additional future organizational growth, which may result in a further processing expansion to 32 minicomputers. This potential growth is 5 to 8 years away and no definite planning is currently being done. It is desired to know if the SPR sized for the 16 ICS system will be able to adequately service a 32 ICS system, and if not, is there one that will.

To determine the approximate job flow rate and traffic intensity of the current system we solve (7) of chapter IV using the BIN algorithm and the system parameters listed above. These values are then used to obtain the current approximate mean SPR response time by applying (13). This results in $\rho_{SPR} = .3508$ and $W_{SPR} = 6.16$. Repeating this procedure for the 4 ICS configuration results in $W_{SPR} = 13.2$. Similarly, the planned 16 ICS configuration yields $W_{SPR} = 475$, and the hypothesized 32 ICS configuration yields $W_{SPR} = 992$. Applying (14) and (16), using the previously computed value of ρ_{SPR} we obtain the β's from which the processing rate of an SPR for each configuration is computed to be:

$u_{SPR}(4) = .338 \leftarrow 74$ msec $\quad \beta(4) = 1.35$,

$u_{SPR}(16) = .865 \leftarrow 29$ msec $\quad \beta(16) = 3.46$, and

$u_{SPR}(32) = 1.57 \leftarrow 16$ msec $\quad \beta(32) = 6.26$.

Processing rates for various secondary storage technologies [HOAGLA 79, TOOMBS 78, WARNAR 79] indicate that an SPR subsystem using existing rotating disk technology can support a 16 ICS configuration. This same SPR subsystem cannot adequately support a 32 ICS configuration, although a faster SPR subsystem using this same technology can support a 32 ICS configuration.

Example 2

For the second example a POS environment is considered. POS can generally be described as many small independent processors, each processing a single job, and accessing a shared inventory data base subsystem (IDBS). The inventory data base may be implemented in several different configurations. One is a distributed configuration, where each geographically distinct organizational unit has its own local IDBS, which is shared among its own POS stations. Another configuration is a centralized one, where a single IDBS is located at a single site and is shared by all the POS stations. There are of course a spectrum of configurations in between these two extremes. The centralized configuration requires additional communication facilities from the IDBS central site to each of the geographically distinct organizational units.

Structurally each POS station consists of a processor and a number of local I/O devices. These I/O devices may include any of the following typical devices; (1) a digital display or two, (2) a printer for a sales receipt, (3) an input scanner, (4) an input alphanumeric keyboard, and (5) an auxiliary input device, for instance a scale.

A normal transaction consists of one or more human interactions to enter data through the I/O devices. The station processor, once it accepts the data, requests service from the IDBS to process this data. This request is serviced on a FCFS basis by the IDBS with an exponentially distributed service time. Any time devoted to communication between a POS station and the IDBS will be incorporated into the IDBS service time. It is assumed that the communication subsystem is not a bottleneck and, therefore, this time may be accounted for by increasing the mean processing time of the IDBS. The processed data is returned to the POS station, which has been idle while waiting, and it is then displayed. The cycle is then repeated.

There is only a single job at each station and the station processor is fast enough to service all of its requests and manage all of its devices. Due to the nature of service requests the service time of the station processor is generally constant for local operations, while the human interaction through the devices is random and is assumed to be exponentially distributed. For a single job there is no need to model each device within a station separately since there is no contention for

devices. Also, since the station processor has essentially a constant service time and is not a bottleneck, the mean time of the human interaction distribution can be increased to account for it. Therefore, it is assumed that all of these devices and the station processor can logically be thought of as a single device with an exponentially distributed service time. A feedback loop to the POS station provides for error conditions that arise, which mainly occur when entering data.

The processing rate of the POS station is normalized to 1. Each transaction requires one or more service requests at the POS station followed by a single service request to the IDBS. Each additional service request to the POS station represents a re-entry of data which was required by an error on the previous attempt. The response as seen by the user is strictly a local one. Therefore, the only time the IDBS affects the response time is when the data entry was successful. Based on this descriptive model, an acceptable response time is on the order of 1/4 sec. The POS mean processing time is 1 sec. Therefore, IDBSs which have processing rates many times faster than the POS station must be considered. The range of potential IDBS processing rates is 5 to 2000 times faster than the POS station. At 2000 times faster, the IDBS would require a mean processing time of .5 msec. This represents the upper limit of current secondary storage subsystem technology [HOAGLA 79, TOOMBS 78, WARNAR 79], especially if a communication subsystem is involved.

The jobs have an exponential service time at each device. The POS station mean service time of a job is 1 sec. For this model the mean service times and their estimated transition probabilities are as follows:

mean service time	estimated transition probabilities
$u_{IDBS} = 5\text{-}2000$	$p_{IDBS} = .90$
$u_{POS} = 1$	$p_{POS} = .10$

It is desired to size the IDBS for the two configurations under consideration. The first configuration being considered is a distributed configuration supporting 20 POS stations. The second configuration is a centralized one supporting 1000 POS stations.

To determine the approximate job flow rate and traffic intensity of a 20 POS station configuration we solve (7) of chapter IV using the BIN algorithm and the system parameters listed above. These values are then used to obtain the corresponding approximate mean IDBS response time by applying (13). This results in ρ_{IDBS} =.6797 and W_{IDBS} =.250 for $u_{IDBS}(20) = 12.5$. Applying (14) and (16), using the previously computed value of ρ_{IDBS} we obtain the β from which the processing rate of an IDBS to support a 1000 POS station configuration is $u_{IDBS}(1000) = \beta\, u_{IDBS} = 428.9$.

The local 20 POS station configuration requires a $u_{IDBS} = 80$ msec, which is a reasonable speed for disk technology. The 1000 POS station configuration requires a $u_{IDBS} = 2.33$ msec. For this configuration the IDBS is remotely located and at this speed the communications delay, although not a bottleneck, must be accounted for. This communications delay has been measured to be .75 msec, resulting in a IDBS with a mean processing time of 1.58 msec. This speed exceeds the current capability of disk technology, but can be met using magnetic bubble technology.

An analysis of a modular expansion was performed using both models. One of the key design aspects is the effect on performance due to a modular expansion, or conversely, the amount of increased capability required by the shared resource to continue to deliver some threshold amount of performance. The exact model yielded only an lower bound for the incremental increase in the SPR processing rate required to maintain system response time. This lower bound was too high to be of any practical use.

The analysis of the approximate model yielded a useful and intuitively satisfying relation between the addition of ICSs and the incremental increase in SPR processing rate required to maintain system response time. It indicated that for each ICS added to expand the system, the required increase in incremental SPR processing rate was directly related to the incremental traffic intensity caused by each additional ICS. The implication is that, for example, by doubling the number of ICSs, an increase in the SPR processing rate of < 2 is required to maintain system response time, since the traffic intensity for a stable system is always < 1. This result is verified when compared to values predicted by the exact model. This result will be useful to designers and analysts when they consider building new systems or augmenting existing systems which are based on this class of architecture. Two examples illustrating the utility of the approximate model were presented.

VI. SUMMARY AND RECOMMENDATIONS

A. Summary

The basis of resource sharing and its application to computer architecture has been discussed. Some examples of architectures that support resource sharing were provided, and many more will be constructed. It was our intention to investigate the performance of this class of computer architecture which shares a single processing resource among multiple independent computing systems through the use of analytic queueing models

Utilizing multi-class queueing network theory and the structure of this class of computer architecture a specific queueing network model was developed. Two efficient computational algorithms, SOP and FAC, were presented which could be used to evaluate the performance measures of this model. Previously the evaluation of queueing network models required memory-space and time complexity both growing exponentially with the size of the state-space, $O[(J_i+1)^R]$. The algorithms developed here to evaluate the model require a memory-space which grows linearly, $O[R(J_i+1)]$, with the size of the state-space, although the time complexity still grows exponentially. This provides the designer and analyst the ability to evaluate this model when it has a large state-space if he or she is willing to invest the computation time. Whereas, previously it may not have been possible to evaluate this model due to memory-space limitations.

Although the algorithms to evaluate this exact queueing network model are memory-space efficient, they are still of exponential time complexity. This computational complexity limits the utility of the model, as is generally the case for other multi-class queueing network models. In an attempt to overcome this computational limitation, an approximate queueing model was introduced. This approximate model consists of a set of independent M/M/1 single server queues. The solution technique is based on approximating the job flow rate between these queues. An efficient algorithm, predicated on a binary search technique, was presented to evaluate the performance measures of this approximate model. The development and solution form presented apply to balanced as well as unbalanced systems. The solution algorithm and the error analysis considered only balanced systems

To determine the utility of this approximate model a comparison of results between it and the exact model was made. A random sample space of the input parameters for these models was generated, and the corresponding performance measures evaluated. The performance measures predicted by this approximate model do result in a varying error, which is considered to be within ac eptable engineering limits. The efficiency gained in evaluation of the performance measures is, for most applications, thought to be an acceptable compromise for the error incurred. For situations with extremely large state-spaces, it may be the only analysis method possible. As a result, designers and analysts are provided the capability to use the approximate model to obtain estimates of the performance of a large number of system confiurations in a very short period of time. Once a small number of candidate configurations are culled, the exact model may be applied to obtain more accurate performance predictions.

An analysis of a modular expansion was performed using both models. One of the key design aspects is the effect on performance due to a modular expansion, or conversely, the amount of increased capability required by the shared resource to continue to deliver some threshold amount of performance. The exact model yielded only a lower bound for the incremental increase in the SPR processing rate required to maintain system response time. This lower bound was too high to be of any practical use.

The analysis of the approximate model yielded a useful and intuitively satisfying relation between the addition of ICSs and the incremental increase in SPR processing rate required to maintain system response time. It indicated that for each ICS added to expand the system, the required increase in incremental SPR processing rate was directly related to the incremental traffic intensity caused by each additional ICS. The implication is that, for example, by doubling the number of ICSs, an increase in the SPR processing rate of < 2 is required to maintain system response time, since the traffic intensity for a stable system is always < 1. This result is verified when compared to values predicted by the exact model. This result will be useful to designers and analysts when they consider building new systems or augmenting existing systems which are based on this class of architecture. Two examples illustrating the utility of the approximate model were presented.

B. Research Extensions

We have stated that our position in accepting this M/M/1 approximation was that it "tracked" the performance measures predicted by the exact models, although the values predicted were in error by a varying degree. Similar approximate models based on other independent queueing systems or a mix of different systems may provide a better "fit" than the M/M/1. The M/M/1 system was chosen because the simplicity of its mathematical formulation provides both an expression that yields insight and one that allows manipulation for analysis.

To provide insight to the designer and analyst as to the error incurred when they apply this approximation an error analysis was presented. This analysis was meant to indicate parameter sensitivity and trends. It was not an elaborate analysis, nor were the results conclusive. Further work is needed to characterize the error incurred over a wider range of the parameter values, especially the job allocation vector, J, and the number of ICSs, R.

The BIN algorithm presented in chapter IV, provided an efficient solution to the approximate model when the system is balanced (identical ICSs). Additional work is needed to establish an efficient algorithm for the general case of an unbalanced system, because this system requires solution of a set of simultaneous nonlinear equations. Existing solution techniques should be investigated, including converging iterative ones as is the BIN algorithm. Also, a similar error analysis should be done to determine if an unbalanced system results in different accuracy patterns or is any differently parameter sensitized than a balanced system.

In certain situations the error incurred by using the M/M/1 approximation is not satisfactory. The alternative is to utilize the exact model, but the computation time required may be excessive. Similar models, based on other queueing systems may provide results with a lower error tolerance and less variation. It would be useful to formulate efficient solution algorithms and perform similar error analyses for them, as was done for the M/M/1 approximation. Assuming these other approximations yield values significantly closer to the exact values, the results of a modular expansion analysis would be useful. The results of this analysis would be interesting to compare to the results obtained from the M/M/1 approximation.

A greatest lower bound for the exact model should be pursued. The one established here is too high to be of any practical value. A value of unity was conjectured for the exact model, but not proven, in this dissertation. Proving either this conjecture or the fractional bound (ρ), estab'ished through the analysis of these approximate models, is a useful endeavor.

There are other research areas that may be pursued based on these modeling techniques. One is formulation of more efficient exact and approximate models for architectures with multiple SPRs. These models would be especially applicable to architectures incorporating a LAN allowing high speed access to multiple shared subsystems.

Another research direction would be to establish non-exponential approximate models. These could be based on M/G/1 servers or M/G/1/K servers. The utility of these models could be established by comparing their predictions to those of Shum's EPF model [SHUM 76, SHUM 77]. The EPF model is an approximation based on a queueing network formulation, in which the product terms are replaced by terms representing M/G/1/K servers. The computation time and memory-space complexity of the EPF model are equivalent to those of general queueing network theory. An approximate non-exponential model of time and space complexity comparable to that of the approximate SCS model would be useful.

APPENDIX A

Review of Balance Equations for Queueing Networks

1. The Single Server Queue

To understand the balance, or flow equations we shall start with a simple single server queue. Assume the service time distribution is exponential, with mean u , and assume the arrival process is Poisson (i.e. has exponentially distributed interarrival intervals), with mean a . Let the probability of n jobs in the queue at time t be

$$Pr[n(t)] = Pr[X(t)=n], \qquad t>0 \quad ,$$

where X(t) is a random variable and n is the state of the system, which is the number of jobs in the queue (including any being serviced). Let us also assume that in a small time interval, Δt, at most only one event can occur. Therefore, the state probability balance equation is

$$\begin{aligned} Pr[n(t+\Delta t)] &= Pr[n(t)]\, Pr[\text{no arrv. + no departures in } \Delta t \,/\, n \text{ at } t] \\ &+ Pr[n(t)\text{-}1]\, Pr[1 \text{ arrv. + no departures in } \Delta t \,/\, n\text{-}1 \text{ at } t] \\ &+ Pr[n(t)+1]\, Pr[\text{no arrv. + 1 departure in } \Delta t \,/\, n+1 \text{ at } t] \\ &+ Pr[\Delta t^{2}] \qquad , \end{aligned}$$

where $Pr[\Delta t^{2}]$ is the collective probability that more than one event occurs during Δt. We make Δt is small enough so that this probability is essentially zero, $Pr[\Delta t^{2}]=0$.

Since the arrival and departure processes are exponential (i.e. memoryless), we are able to remove the conditions in the above equation which yields

$$\begin{aligned} Pr[n(t+\Delta t)] &= Pr[n(t)]\, Pr[\text{no arrv. + no departures in } \Delta t] \\ &+ Pr[n(t)\text{-}1]\, Pr[1 \text{ arrv. + no departures in } \Delta t] \\ &+ Pr[n(t)+1]\, Pr[\text{no arrv. + 1 departure in } \Delta t] \qquad . \end{aligned} \tag{1}$$

By definition the probability of each process occuring is

$$\Pr[\text{an arrival occurs in } \Delta t] = 1 - e^{(-a\Delta t)} \quad , \text{ and}$$

$$\Pr[\text{a departure occurs in } \Delta t] = 1 - e^{(-u\Delta t)} \quad .$$

Expanding the exponential by an infinite series and truncating after the second term yields

$$e^{-a\Delta t} = 1 - a\Delta t + \frac{(a\Delta t)^2}{2!} - \frac{(a\Delta t)^3}{3!} + \ldots \approx 1 - a\Delta t \quad .$$

This truncation may occur since for small Δt, $\Delta t^2 << \Delta t$, and as Δt approaches zero in the limit, Δt^2 approaches zero much more rapidly. Therefore, all the higher order terms are negligible compared to the first order term. Substituting this back into the probability definitions we obtain the following approximations

$$\Pr[\text{an arrival occurs in } \Delta t] \approx a\Delta t \quad , \text{ and}$$

$$\Pr[\text{a departure occurs in } \Delta t] \approx u\Delta t \quad ,$$

and conversely

$$\Pr[\text{no arrivals occur in } \Delta t] \approx 1 - a\Delta t \quad , \text{ and}$$

$$\Pr[\text{no departures occur in } \Delta t] \approx 1 - u\Delta t \, .$$

Noting that the arrival and departure processes are independent of each other allows us to express their joint probability as a product of the two probabilities, which as above may be approximated by the first order term, as

$$\begin{aligned} \Pr[\text{no arriv. + no departures in } \Delta t] &= (1-a\Delta t)(1-u\Delta t) \\ &= 1 - a\Delta t - u\Delta t + au(\Delta t)^2 \\ &\approx 1 - (a+u)\Delta t \quad . \end{aligned}$$

Applying this truncation again allows us to express the following probabilities as

$$\begin{aligned} \Pr[\text{1 arriv. + no departures in } \Delta t] &= a\Delta t\,(1-u\Delta t) \\ &= a\Delta t - a\Delta t(\Delta t)^2 \\ &\approx a\Delta t \quad , \text{ and} \end{aligned}$$

$$\Pr[\text{no arriv. + 1 departure in } \Delta t] = (1-a\Delta t)\, u\Delta t$$
$$= u\Delta t - au(\Delta t)^2$$
$$\approx u\Delta t \quad .$$

Substituting these results into equation (1) yields

(2) $$\Pr[n(t+\Delta t)] = \Pr[n(t)]\,(1-(a+u)\Delta t) + \Pr[n(t)-1]\,a\Delta t + \Pr[n(t)+1]\,u\Delta t$$

Assuming an infinite size queue and, therefore, no upper limit on the number of jobs, a lower limit must still be considered since it is physically impossible to have less than an empty queue. This results in the following boundary equation for an empty queue (i.e. state = "0")

(3) $$\Pr[0(t+\Delta t)] = \Pr[0(t)]\,(1-a\Delta t) + \Pr[1(t)]\,u\Delta t \quad .$$

When the queue is empty no departures can occur and there exists no state = "-1". Rearranging equations (2) and (3) algebraically yields

$$\Pr[n(t+\Delta t)] = \Pr[n(t)+1]\,u\Delta t - \Pr[n(t)]\,(a+u)\Delta t + \Pr[n(t)-1]\,a\Delta t + \Pr[n(t)] \quad , \text{ and}$$

$$\Pr[0(t+\Delta t)] = \Pr[1(t)]\,u\Delta t - \Pr[0(t)]\,a\Delta t + \Pr[0(t)] \quad .$$

Then dividing by Δt and taking Δt to its limit results in

(4) $$\lim_{\Delta t \to 0} \frac{\Pr[n(t+\Delta t)] - \Pr[n(t)]}{\Delta t} = \Pr[n(t)+1]\,u - \Pr[n(t)]\,(a+u) + \Pr[n(t)-1]\,a \quad , \text{ and}$$

(5) $$\lim_{\Delta t \to 0} \frac{\Pr[0(t+\Delta t)] - \Pr[0(t)]}{\Delta t} = \Pr[1(t)]\,u - \Pr[0(t)]a \quad .$$

Note that these equations are the derivatives (Pr'[n], etc.) of the state probabilities (that is, the state probabilistic rate of change). Assuming a stationary distribution implies

$$\Pr[n(t)] = \Pr[n] \quad .$$

The average rate of change must be zero, or an infinite accumulation in some state might occur. Therefore, equations (4) and (5) are assumed to be identically zero, resulting in

$$Pr'[n] = 0 = Pr[n+1]\,u - Pr[n]\,(a+u) + Pr[n-1]\,a \quad , \text{ and}$$

$$Pr'[0] = 0 = Pr[1]\,u - Pr[0]\,a \quad .$$

By rearranging these equations we obtain

(6) $$Pr[n]\,(a+u) = Pr[n+1]\,u + Pr[n-1]\,a \quad , \text{ and}$$

(7) $$Pr[0]\,a = Pr[1]\,u \quad .$$

From studying these equations we realize that the left side is the (flow) rate out of a state, while the right side is the rate into that state. Therefore, these equations represent the balance of flow between states, hence the term balance (or flow) equations.

2. Open and Closed Queueing Networks

To obtain the balance equations for a network of queues (i.e. a Jacksonian network [JACKSO 63]) the same basic procedures are necessary. We now have an additional problem of many independent queues connected to each other. This changes our scalar state n for one queue, to a vector $N=(n_0, \ldots, n_s)$ to account for each of the $s+1$ queues. The state equilibrium probability distribution becomes

$$Pr[N] = Pr[n_0, \ldots, n_s]$$

where

$$K = \sum_{i=0}^{s} n_i$$

and n_i is the number of jobs in the i-th queue, and K is the total number of jobs in the network. Similarly, the mean arrival and departure rates become vectors also, one element for each queue, a_i and u_i. To specify how these queues are connected, we use constant transition probabilities to indicate the flow (transition) of one job from queue j to queue i, $p_{j,i}$. Without proceeding through

the tedious algebra (as in the previous section) we may write [JACKSO 63] the left side as

$$\left(\sum_{i=0}^{s} a_i + \sum_{i=0}^{s} u_i \, y(n_i) \right) \Pr[N]$$

where

$$y(n_i) = \begin{cases} 0 & , \; n_i \leq 0 \\ 1 & , \; n_i > 0 \end{cases}$$

Note that the extra factor $y(n_i)$ is equivalent to accounting for our "less than empty" boundary conditions, since some queues may be empty while others are not. The right side is a little more complex. We are looking for all the ways to reach the destination state N through a single transition from a source state, N'. Any queue i may receive a job from any other queue j, therefore, the source state must be of the form $N' = (n_0, \ldots, n_j+1, \ldots, n_i-1, \ldots, n_s)$. All possible combinations of source states of this form must be accounted for on the right side of the equation, which yeilds

$$\sum_{j=0}^{s} \sum_{i=0}^{s} u_j \, y(n_i) \, p_{j,i} \Pr[n_0, \ldots, n_j+1, \ldots, n_i-1, \ldots, n_s]$$

$$+ \sum_{i=0}^{s} a_i \, y(n_i) \Pr[n_0, \ldots, n_i-1, \ldots, n_s] \quad .$$

These portions yield the following for an open queueing network with exponential service and interarrival times, and an infinite queueing capacity [JACKSO 63]:

(8)
$$\left[\left(\sum_{i=0}^{s} a_i \right) + \left(\sum_{i=0}^{s} u_i \, y(n_i) \right) \right] \Pr[n_0, \ldots, n_s]$$

$$= \left[\sum_{i=0}^{s} \sum_{j=0}^{s} u_j \, p_{j,i} \, y(n_i) \Pr[n_0, \ldots, n_j+1, \ldots, n_i-1, \ldots, n_s] \right.$$

$$\left. + \left[\sum_{i=0}^{s} a_i \, y(n_i) \Pr[n_0, \ldots, n_i-1, \ldots, n_s] \right] \right] \quad .$$

For a closed queueing network with a constant number of jobs, K, continually circulating [GORDON 67] there is no arrival process, which implies $a_i = 0$, for all i. The resulting equation is

$$(9) \qquad \left[\sum_{i=0}^{s} u_i \, y(n_i) \right] \Pr[n_0, \ \ldots \ , n_s]$$

$$= \sum_{i=0}^{s} \sum_{j=0}^{s} u_j \, p_{j,i} \, y(n_i) \Pr[n_0, \ \ldots \ , n_j+1, \ \ldots \ , n_i-1, \ \ldots \ , n_s] \quad .$$

It should be noted that our approach to arrive at equations (8) and (9) was an intuitive extension of equations (6) and (7), the balance equations for a single server queue. A more rigorous approach would require a procedure similar to that from which equations (6) and (7) were derived [JACKSO 57, JACKSO 63, GORDON 67].

The solution to these equations is based on assuming a product form solution [GORDON 67] of the form

$$(10a) \qquad \Pr[n_0, \ \ldots \ , n_s] = \frac{1}{G(K)} \prod_{k=0}^{s} x_k^{n_k} \quad , \text{ and}$$

$$(10b) \qquad \Pr[n_0, \ \ldots \ , n_j+1, \ \ldots \ , n_i-1, \ \ldots \ , n_s] = \frac{x_j / x_i}{G(K)} \prod_{k=0}^{s} x_k^{n_k} \quad ,$$

where G(K) is a probability normalization factor and the x_k's are unknowns whose solution must be obtained. Substituting the solution form of (10) back into (9) yields

$$\sum_{i=0}^{s} u_i \, y(n_i) \frac{1}{G(K)} \prod_{k=0}^{s} x_k^{n_k} = \left[\sum_{i=0}^{s} \sum_{j=0}^{s} u_j \, p_{j,i} \, y(n_i) \frac{x_j}{x_i} \right] \frac{1}{G(K)} \prod_{k=0}^{s} x_k^{n_k}$$

The term on the right-hand side of the equation, outside the brackets, is always non zero and is a term common to both sides. This term is not dependent on the indices i or j and, therefore, may be manipulated without effecting the summations. Extracting and cancelling this term results in

$$\sum_{i=0}^{s} u_i\, y(n_i) = \sum_{i=0}^{s} \sum_{j=0}^{s} u_j\, p_{j,i}\, y(n_i)\frac{x_j}{x_i} \quad .$$

Further extracting the common term $y(n_i)$ yields

$$(11) \qquad \sum_{i=0}^{s} y(n_i) \left[u_i - \sum_{j=0}^{s} u_j\, p_{j,i} \frac{x_j}{x_i} \right] = 0 \quad .$$

The next step in the solution of this closed queueing network is based on the fact that at any time it is possible for all but one queue to be empty. Let this single non-empty queue be the k-th queue. In this case

$$y(n_i) = \begin{cases} 0 & , \quad i \neq k \\ 1 & , \quad i = k \end{cases} \quad .$$

Hence, (11) is reduced to

$$u_i - \sum_{j=0}^{s} u_j\, p_{j,i} \frac{x_j}{x_i} = 0 \quad , \qquad \text{for } i = 0, \ldots, s \quad .$$

By rearranging the above the following results are obtained:

$$x_i u_i - \sum_{j=0}^{s} x_j u_j\, p_{j,i} = 0 \quad .$$

Letting $e_i = x_i u_i$; then

$$(12) \qquad e_i = \sum_{j=0}^{s} e_j\, p_{j,i} \quad ,$$

which is the vector equation E = E[P], where the vector $E=(e_0, \ldots, e_s)$ and the matrix $[P]=[p_{j,i}]$. The e_i's are usually called the relative visist frequencies. Finally we can solve for the probability normalization factor, by using the fact that the probabilities must sum to unity. Hence, from (10) we may formulate

$$\sum_N Pr[n_0, \ldots, n_s] = 1 = \frac{1}{G(K)} \sum_N \prod_{i=0}^{s} x_i^{n_i} \quad .$$

Therefore,

$$G(K) = \sum_N \prod_{i=0}^{s} x_i^{n_i} \quad , \tag{13}$$

where the summation over N implies all non-negative solutions to the equation

$$\sum_{i=0}^{s} n_i = K \quad .$$

See [KLIENR 76], pg216.

In summary, the state equilibrium probabilities can be obtained from the following

$$Pr[n_0, \ldots, n_s] = \frac{1}{G(K)} \prod_{i=0}^{s} x_i^{n_i} \quad , \text{ where}$$

$$G(K) = \sum_N \prod_{i=0}^{s} x_i^{n_i} \quad .$$

3. The Central Server Model

A simple, nontrival case of a closed queueing network is called the Central Server Model [BUZEN 71]. A diagram of this model is shown in figure A-1 and the corresponding transition probability

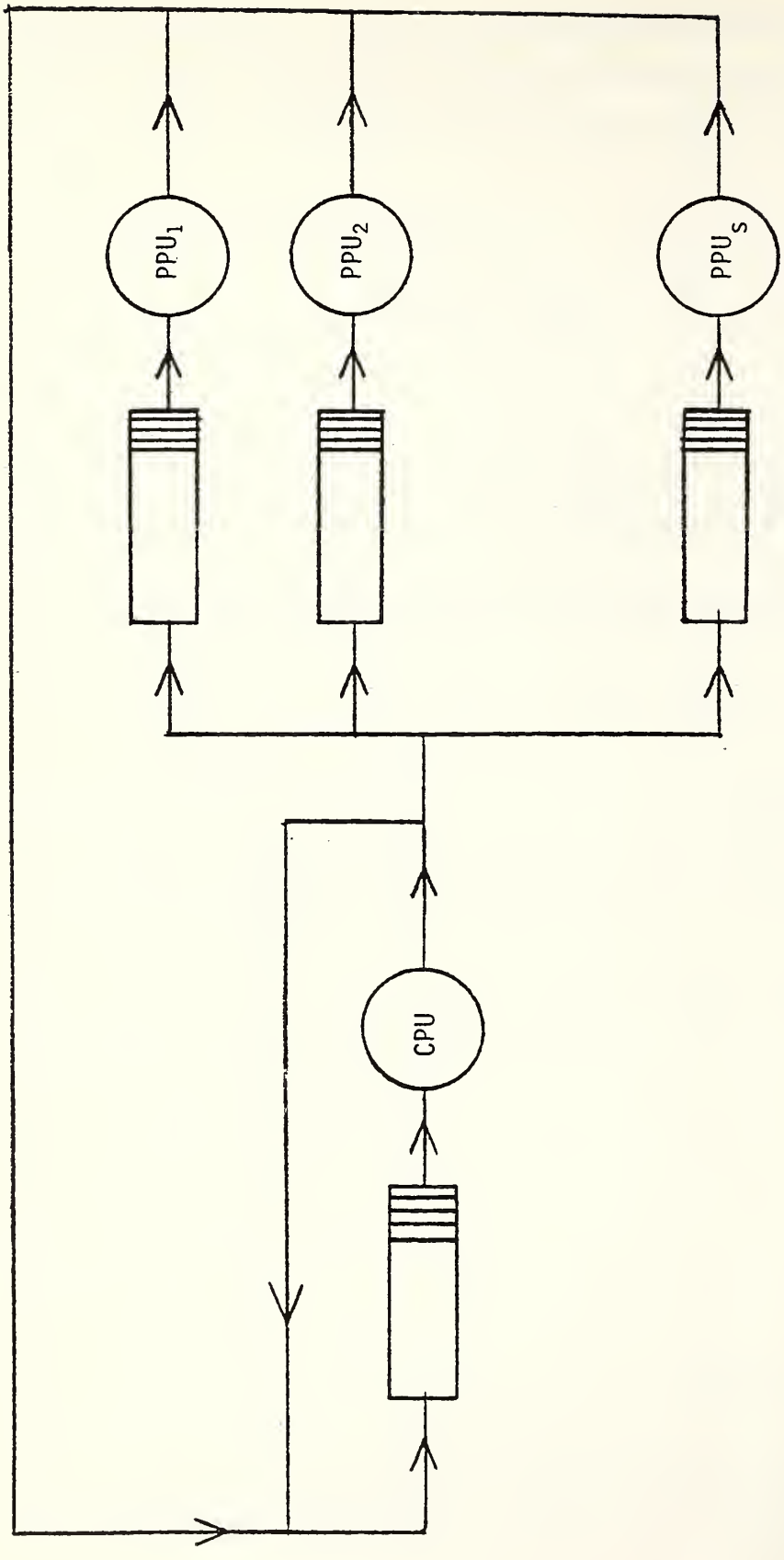

Figure A-1. Central server model block diagram.

$$[P] = \begin{bmatrix} p_{0,0} & p_{0,1} & p_{0,2} & \cdots & p_{0,s} \\ 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ & \cdot & & \cdot & \cdot \\ & \cdot & & \cdot & \cdot \\ & \cdot & & \cdot & \cdot \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

Figure A-2. Central Server model transition probability matrix.

matrix is structured as shown in figure A-2. This results in a simple solution to (12), which consists of $s+1$ simultaneous equations with $s+1$ unknowns. In this case there are only s independent equations. Therefore, one can solve all these equations in terms of one of the unknowns, say e_0. This does not result in a unique (absolute) solution, but rather a relative solution. Consequently, any value substituted for e_0 will yield a solution satisfying (12). Therefore, this relative solution is related to the absolute solution by some multiplicative constant. The relative solution for the Central Server Model is

$$E = (e_0, e_0 p_{0,1}, \ldots, e_0 p_{0,s}) \quad .$$

Note that the first subscript (j) of the $p_{j,i}$'s denotes the source of the transition. There is only a single source in this model, the central server. Therefore, this subscript may be dropped since it is implied, resulting in

$$E = (e_0, e_0 p_1, \ldots, e_0 p_s) \quad .$$

Letting $x_0 = 1$, which then implies $e_0 = u_0 x_0 = u_0$ results in

$$E = (u_0, u_0 p_1, \ldots, u_0 p_s) \quad ,$$

and

$$X = (1, \frac{u_0 p_1}{u_1}, \ldots, \frac{u_0 p_s}{u_s}) \quad .$$

The structure of the central server model results in a simple solution to (12). The evaluation of (13), although straightforward, requires a summation over a state-space whose size increases exponentially, $O[(K+1)^s]$. Buzen [BUZEN 71] developed an iterative method for evaluating (13) has requires a computational complexity $O[Ks]$, vs. $O[(K+1)^s]$. This method is based on an recursive partitioning technique. Define the following auxiliary function

$$(14) \qquad g(m,j) = \sum_{\substack{\sum_{i=0}^{j} n_i = m}} \prod_{i=0}^{j} x_i^{n_i}$$

Note that when $m=K$ and $j=s$, then $G(K)=g(K,s)$. By partitioning $g(m,j)$ based on the occupancy of the last queue, j, being either empty or not, yields

$$(15) \quad g(m,j) = \sum_{\substack{\sum_{i=0}^{j-1} n_i = m}} \prod_{i=0}^{j-1} x_i^{n_i} + x_j \sum_{\substack{\sum_{i=0}^{j} n_i = m-1}} \prod_{i=0}^{j} x_i^{n_i} = g(m,j-1) + x_j\, g(m-1,j) \quad ,$$

with the following boundary conditions

$$g(0,j) = 1 \quad , \quad 0 \le j \le s \quad , \quad \text{and}$$

$$g(k,0) = x_0^{k} \quad , \quad 0 \le k \le K \, .$$

Equation (14) can be evaluated for any values of m and j using the boundary conditions above, or any other. By conceptualizing g(m,j) as a matrix, Buzen presents an efficient iterative algorithm for evaluating (14). As a result, the normalization constant (13) and expressions based on it can be evaluated much more efficiently than was previously possible.

Buzen also derived expressions for some performance measures which are based on a similar product form structure and, therefore, may be efficiently evaluated by the above iterative technique. These performance measures include the device busy probabilities and the mean queue length for the central server model. The device busy probability, A_j, is the probability that the j-th queue is not empty and can be expressed as

$$A_j = \sum_{N(\exists n_j \ge 1)} Pr[n_0, \ldots, n_s] = \sum_{\substack{\sum_{i=0}^{s} n_i = K \; (\exists n_j \ge 1)}} \left[\frac{1}{G(K)} \prod_{i=0}^{s} x_i^{n_i} \right] = \frac{x_j}{G(K)} \sum_{\substack{\sum_{i=0}^{s} n_i = K-1}} \prod_{i=0}^{s} x_i^{n_i}$$

$$= \frac{x_j}{G(K)} \, G(K-1)$$

To derive the mean queue length of the j-th queue, Q_j, first define $R_j(k)$ as the probability that the j-th queue has k or more jobs (i.e. $n_j \geq k-1$), expressed as

$$R_j(k) = \sum_{N(\exists n_j \geq k)} Pr[n_0, \ldots, n_s] = \sum_{\substack{\sum_{i=0}^{s} n_i = K \\ (\exists n_j \geq k)}} \left[\frac{1}{G(K)} \prod_{i=0}^{s} x_i^{n_i} \right] = \frac{x_j^k}{G(K)} \sum_{\sum_{i=0}^{s} n_i = K-k} \prod_{i=0}^{s} x_i^{n_i}$$

Hence,

$$R_j(k) = \frac{x_j^k}{G(K)} G(K-k)$$

Note that $R_j(0)=1$, $R_j(1)=A_j$, $R_j(K+1)=0$, and the probability that the j-th queue has exactly k jobs is $R_j(k)-R_j(k+1)$. Therefore, the mean queue length is

$$Q_j = \sum_{k=1}^{K} k [R_j(k)-R_j(k+1)] = \sum_{k=1}^{K} k R_j(k) - \sum_{k=1}^{K} k R_j(k+1)$$

$$= \sum_{k=1}^{K} k R_j(k) - [\sum_{k=1}^{K} (k-1) R_j(k) + K R_j(K+1)]$$

$$= R_j(1) + \sum_{k=2}^{K} k R_j(k) - \sum_{k=2}^{K} (k-1) R_j(k)$$

$$= R_j(1) + \sum_{k=2}^{K} [k-(k-1)] R_j(k)$$

$$= \sum_{k=1}^{K} R_j(k)$$

$$= \sum_{k=1}^{K} \frac{x_j^k G(K-k)}{G(K)}$$

4. Closed Queueing Networks with Multiple Job Classes

Basket et. al. [BASKET 75] have extended the theory of queueing networks to include multiple classes of jobs, with other than exponential service time probability distributions. This extended model is based on expanding the state description, the state transition probabilities, and the mean service rates to include class distinction. Define the following (Note: these definitions differ somewhat from those used in the rest of this dissertation, see Appendix C, but they conform to the definitions generally associated with multi-class queueing network theory):

$N = (n_0, \ldots, n_s)$ is the state-space vector description ;
$= (N_0, \ldots, N_s)$
$= (n_{0,1}, \ldots, n_{0,R}, \ldots, n_{s,1}, \ldots, n_{s,R})$

$n_{i,r}$ = number of class r jobs at the i-th service center (both in service and in queue) ;

$n_i = \sum_{r=1}^{R} n_{i,r}$, the total number of jobs (of all classes) at the i-th service center ;

$N_i = (n_{i,1}, \ldots, n_{i,R})$ is the job class distribution at the i-th service center ;

$K = \sum_{i=0}^{s} n_i = \sum_{i=0}^{s} \sum_{r=1}^{R} n_{i,r}$, the total number of jobs in the network ;

$J = (J_1, \ldots, J_R)$ is the job class allocation vector for the network ;

J_r = the maximum number of jobs that can be allocated to the r-th class ,

$J_r = \sum_{i=0}^{s} n_{i,r}$ is the total number of class r jobs in the network ;

$u_{i,r}$ = the mean service rate of a class r job at the i-th service center ;

$p_{i,r:j,s}$ = the probability of a class r job at the i-th service center transiting to the j-th service center and becoming a class s job ;

R = the number of different job classes in the network

The state equilibrium probabilities are assumed to have the form

$$\Pr[n_{0,1}, \ldots, n_{s,R}] = \frac{1}{G(J)} \prod_{i=0}^{s} f_i(N_i) \quad ,$$

where

the product factors are given by

$$f_i(N_i) = \begin{cases} n_i! \prod_{r=1}^{R} \frac{x_{i,r}^{n_{i,r}}}{n_{i,r}!} & \text{PS, FCFS, LCFS} \\ \prod_{r=1}^{R} \frac{x_{i,r}^{n_{i,r}}}{n_{i,r}!} & \text{IS} \quad ; \end{cases}$$

the normalization constant is given by

$$G(J) = \sum_{N} \prod_{i=0}^{s} f_i(N_i) \quad ,$$

where the summation over the state-space N means all non negative solutions to

$$\sum_{i=0}^{s} N_i = J$$

The relative visit frequency, $e_{i,r}$, is given by the solution to

$$e_{i,r} = \sum_{j=0}^{s} \sum_{t=1}^{R} e_{j,t} \, p_{j,t;i,r} \quad ,$$

and the relative load factor, $x_{i,r}$, is given by

$$x_{i,r} = \frac{e_{i,r}}{u_{i,r}}$$

Buzen's efficient computational procedures have been generalized by Muntz and Wong [MUNTZ 74], Giammo [GIAMMO 76], and Shum [SHUM 77] for networks with a constant number of jobs in each of the R classes, specified by the vector J. The efficient iterative solution technique of (15) for a single class of jobs is based on a two dimensional conceptualization, jobs (a scalar) by devices. A direct extension of this two dimensional technique for multiple classes of jobs requires the substitution of a job class distribution vector for the previous scalar job specification. Defining an auxiliary function, as in (14), results in

$$(16) \qquad g(M;k) = \sum_{\substack{\sum_{i=0}^{k} N_i = M}} \prod_{i=0}^{k} n_i! \prod_{r=1}^{R} \frac{x_{i,r}^{n_{i,r}}}{n_{i,r}!}$$

Following a recursive partitioning procedure similar to that used to obtain (15), yields the following recursive defining relation for the auxiliary function [SHUM 76]

$$(17) \qquad g(M;k) = g(M;k-1) + \sum_{r=1}^{R} x_{i,r}\, g(M-d_r;k) \quad .$$

Equation (16) may also be expressed as [MUNTZ 74]

$$(18) \qquad g(M;k) = \sum_{n_{k,1}=0}^{m_1} \cdots \sum_{n_{k,R}=0}^{m_R} f_k(N_k - M)\; g(M;k-1) \quad ,$$

where M is a vector representing a job class distribution such that

$$M = (m_1, \ \ldots \ , m_R) \quad , \quad m_i = \text{the number of class i jobs} \quad \ni \ m_i \leq J_i \quad ,$$

and d_r is a difference vector such that

$$d_r = (b_1, \ \ldots \ , b_R) \quad , \quad b_i = \begin{cases} 0 & i \neq r \\ 1 & i = r \end{cases} \quad , \ i = 1, \ldots, R \ .$$

The boundary conditions for this auxiliary function are

$$g(0;k) = 1 \quad , \ 0 \leq k \leq s \quad , \text{ and}$$

$$g(M;0) = f_0(M) \quad , \ 0 \leq M \leq J \quad .$$

The probability normalization constant is obtained by summing over the entire state space, which can be expressed in terms of the auxiliary function as [MUNTZ 74]

$$(19) \quad G(J) = \sum_{n_{s,1}=0}^{J_1} \ldots \sum_{n_{s,R}=0}^{J_R} \left[n_s! \prod_{r=1}^{R} \frac{x_{s,r}^{n_{s,r}}}{n_{s,r}!} \left\{ \sum_{\substack{i=0 \\ \sum_{i=0}^{s-1} N_i = J - N_s}}^{s-1} \prod_{r=1}^{R} n_i! \frac{x_{i,r}^{n_{i,r}}}{n_{i,r}!} \right\} \right]$$

$$= \sum_{n_{s,1}=0}^{J_1} \ldots \sum_{n_{s,R}=0}^{J_R} \left[f_s(N_s) \ \{ \ g(J-N_s;s-1) \ \} \right]$$

$$= g(J;s)$$

$$= G(J) \quad ,$$

where

$$g(N_0;0) = f_0(N_0) \quad ,$$

$$f_i(N_i) = \sum_{r=1}^{R} x_{i,r}\, f_i(N_i - d_r) \quad , \text{ and}$$

$$f_i(0) = 1 \quad .$$

This completes the review of the basic equations for current queueing network theory.

The following is provided to demonstrate the details of the partitioning procedure used to obtain (17). We will only use a two queue system to minimize the length of the presentation, while still demonstrating the basic concepts. First, we expand the function definition of (16) to obtain the recursive relation. Then as an alternative approach we expand the right-hand side of (17) by substituting (18) , and show that the left-hand side of (17) is obtained.

From (16) we obtain

$$g(N_0;0) = f_0(N_0) = n_0 ! \prod_{r=1}^{R} \frac{x_{0,r}^{n_{0,r}}}{n_{0,r} !}$$

$$= \sum_{r=1}^{R} x_{0,r} \left[(n_0 - 1) ! \prod_{t=1}^{R} \frac{x_{0,t}^{(n_{0,t} - d_r)}}{(n_{0,t} - d_r) !} \right]$$

Therefore,

$$g(N_0;0) = \sum_{r=1}^{R} x_{0,r}\, g(N_0 - d_r;0)$$

From (18), substituting this definition, we also obtain

$$g(M;1) = \sum_{n_{1,1}=0}^{m_1} \cdots \sum_{n_{1,R}=0}^{m_R} \left\{ n_1! \prod_{r=1}^{R} \frac{x_{1,r}^{n_{1,r}}}{n_{1,r}!} \right\} \left\{ \sum_{\substack{0 \\ \sum N_i = M-N_1 \\ i=0}} \prod_{i=0}^{0} n_i! \prod_{r=1}^{R} \frac{x_{i,r}^{n_{1,r}}}{n_{1,r}!} \right\}$$

$$= \sum_{n_{1,1}=0}^{m_1} \cdots \sum_{n_{1,R}=0}^{m_R} f_1(N_1)\, g(M-N_1;0) \quad ,$$

and

$$g(M-d_r;1) = \sum_{n_{1,1}=0}^{m_1} \cdots \sum_{n_{1,r}=0}^{m_r-1} \cdots \sum_{n_{1,R}=0}^{m_R} n_1! \prod_{s=1}^{R} \frac{x_{1,s}^{n_{1,s}}}{n_{1,s}!}\, g(M-d_r-N_1;0)$$

$$= \sum_{n_{1,1}=0}^{m_1} \cdots \sum_{n_{1,r}=1}^{m_r} \cdots \sum_{n_{1,R}=0}^{m_R} (n_1-1)! \prod_{s=1}^{R} \frac{x_{1,s}^{(d_s n_{1,s}-d_r)}}{(d_s n_{1,s}-d_r)!}\, g(M-N_1;0)$$

$$= \sum_{n_{1,1}=0}^{m_1} \cdots \sum_{n_{1,r}=1}^{m_r} \cdots \sum_{n_{1,R}=0}^{m_R} f_1(N_1-d_r)\, g(M-N_1;0)$$

Restructuring the above yields

$$g(M;1) = f_1(0)\, g(M;0) + \sum_{0<N_1<M} f_1(N_1)\, g(M-N_1;0)$$

$$= g(M;0) + \sum_{0<N_1<M} \left[\sum_{r=1}^{R} x_{1,r}\, f_1(N_1-d_r) \right] g(M-N_1;0)$$

$$= g(M;0) + \sum_{r=1}^{R} \left[\sum_{n_{1,1}=0}^{m_1} \cdots \sum_{n_{1,r}=1}^{m_r} \cdots \sum_{n_{1,R}=0}^{m_R} x_{1,r}\, f_1(N_1-d_r)\, g(M-N_1;0) \right]$$

$$= g(M;0) + \sum_{r=1}^{R} x_{1,r} \left[\sum_{n_{1,1}=0}^{m_1} \cdots \sum_{n_{1,r}=1}^{m_r} \cdots \sum_{n_{1,R}=0}^{m_R} f_1(N_1-d_r)\ g(M-N_1;0) \right]$$

$$= g(M;0) + \sum_{r=1}^{R} x_{1,r}\ g(M-d_r;1) \quad ,$$

which is also

$$g(M;1) = \left\{ f_1(M)\, g(\underline{0};0) + f_1(M-d_r)\, g(\underline{0}+d_r;0) + \cdots + f_1(\underline{0}+d_r) g(M-d_r;0) + f_1(\underline{0})\, g(M;0) \right\}$$

$$= \left\{ f_1(M)\, g(\underline{0};0) + f_1(M-d_r)\, g(\underline{0}+d_r;0) + \cdots + f_1(\underline{0}+d_r) g(M-d_r;0) \right\} + f_1(\underline{0})\, g(M;0)$$

$$= g(M;0) + \sum_{r=1}^{R} x_{1,r}\ g(M-d_r;1)$$

As an alternative approach we will expand the right-hand side of (17), substituting (18) and derive the equality of (17).

$$\sum_{r=1}^{R} x_{1,r}\ g(M-d_r;1) = \sum_{r=1}^{R} x_{1,r} \left\{ \sum_{n_{1,1}=0}^{m_1} \cdots \sum_{n_{1,r}=1}^{m_r} \cdots \sum_{n_{1,R}=0}^{m_R} f_1(N_1-d_r)\ g(M-N_1;0) \right\}$$

$$= \sum_{r=1}^{R} x_{1,r} \left\{ f_1(M-d_r)\, g(\underline{0};0) + \cdots + f_1(\underline{0})\, g(M-d_r;0) \right\}$$

$$= \left\{ f_1(M)\, g(\underline{0};0) \;+ f_1(M\text{-}d_r)\, g(\underline{0}+d_r;0) \;+ \ldots \;+ f_1(\underline{0}+d_r) g(M\text{-}d_r;0) \right\} ,$$

and by adding the final term to complete the series and by noting that

$$g(M;\underline{0}) \;= f_1(\underline{0})\, g(M;0) \qquad , \text{ and}$$

$$f_1(\underline{0}) \;= 1$$

we obtain

$$g(M;0) \;+ \sum_{r=1}^{R} x_{1,r}\, g(M\text{-}d_r;1) \;= \left\{ f_1(M)\, g(\underline{0};0) \;+ f_1(M\text{-}d_r)\, g(\underline{0}+d_r;0) \right.$$

$$\left. + \ldots \;+ f_1(\underline{0}+d_r) g(M\text{-}d_r;0) \right\} + g(M;0)$$

$$= g(M;1)$$

Following the above procedure it can be shown that the general case yields

$$g(M;k) \;= g(M;k\text{-}1) \;+ \sum_{r=1}^{R} x_{i,r}\; g(M\text{-}d_r;k)$$

APPENDIX B

GLOSSARY

ASSIGN	the assignment algorithm
BIN	the BIN algorithm
balanced system	each ICS is identical
C	coefficient of variation
EPF	extended product form model
FAC	factorial expansion algorithm
FCFS	first-come first-served scheduling policy
ICS	independent computing system
IDBS	inventory data base subsystem
IS	scheduling policy consisting of an infinite number of servers, i.e. no scheduling policy
iid	statistically independent and indentically distributed
LAN	local area network
LCFS	last-come-first-served-preemptive-resume scheduling policy
LSI	large scale (circuit) integration
M/G/1	a single server queue with Poisson arrival and general service time distribution

M/G/1/K	a single server queue with Poisson arrival and general service time distribution, with a queueing limit of K jobs
M/M/1	a single server queue with Poisson arrival and exponential service time distribution
M/M/1/K	a single server queue with Poisson arrival and exponential service time distribution, with a queueing limit of K jobs
POS	point-of-sales
PPU	peripheral processing unit
PS	processor sharing scheduling policy
Relative error	(Exact value - Approximate value)/(Exact value)
SCS	Shared Central Server
SOP	sum-of-products expansion algorithm
SPR	shared processing resource

APPENDIX C

MATHEMATICAL NOTATION

$a_{i,j}$ mean arrival rate for device (i,j)

$A_{i,j}$ the busy probability of device (i,j)

β assumed SPR increase processing rate factor

$e_{i,j} = \sum_{t=0}^{L-1} \sum_{r=1}^{R} e_{t,r}\, p_{t,r;i,j}$ relative visit frequency for device (i,j)

$d_i = (b_1, \ldots, b_R)$ a unit vector, $\ni \; b_r = \begin{cases} 0 & r \neq i \\ 1 & r = i, \end{cases}$, $r = 1, \ldots, R$

$f_i(N_i) = \begin{cases} n_i! \prod_{r=1}^{R} \dfrac{x_{i,r}^{n_{i,r}}}{n_{i,r}!} & \text{PS, FCFS, LCFS} \\ \prod_{r=1}^{R} \dfrac{x_{i,r}^{n_{i,r}}}{n_{i,r}!} & \text{IS} \end{cases}$ product factors

$G(K) = G(J)$ normalization constant

$g(M;k)$ auxiliary iterative function used in the computation of the normalization constant

$h(i,J_i)$ aggregate of the auxiliary iterative function

i,j index denoting device j within the i-th ICS

$J = (J_1, \ldots, J_R)$ is the job class allocation vector for the network

$J_i = \sum_{r=1}^{s_i} n_{i,r}$ the maximum number of jobs that can be allocated to the i-th class (i-th ICS)

$J_r = J_i$ for i and $r = 1, \ldots, R$ when the network is balanced, and in Chapter IV J is used as a generic scalar such that $J = J_i$

$K = \sum_{i=1}^{R} J_i = \sum_{i=1}^{R} \sum_{r=0}^{s_i} n_{i,r}$ the total number of jobs in the network

$K = RJ_i$, $i = 1, \ldots, R$ when the network is balanced

$L = \sum_{i=0}^{R} s_i$ total number of devices in the network

$M = (m_1, \ldots, m_R)$ is a dummy counting vector that may range over the job allocation vector , $\ni\ 0 \leq m_i \leq J_i$, $\|M\| = \prod_{i=1}^{R} (m_i + 1)$, and $|M| = \sum_{i=1}^{R} m_i$

$\|M\| = \prod_{i=1}^{R} (m_i + 1)$ is the vector range (product)

$|M| = \sum_{i=1}^{R} m_i$ is the vector value (summation)

$n = (n_0, \ldots, n_s)$ state vector description of the network
$= (n_{1,0}, \ldots, n_{1,s_1}, \ldots, n_{R,0}, \ldots, n_{R,s_R})$

$n_{i,r}$ number of class i jobs at the r-th service center of the i-th ICS (both in service and in queue)

n_i the total number of jobs (of all classes) at the i-th ICS or the SPR

$$n_i = \begin{cases} \sum_{r=1}^{R} n_{i,r} & i>0 \\ \sum_{r=1}^{R} n_{r,0} & i=0 \end{cases}$$

$N_0 = (n_{1,0}, \ldots, n_{R,0})$ is the job class distribution at the SPR

$O[X]$ the order of complexity of X

$Pr[n_{1,0}, \ldots, n_{R,s_R}]$ state equilibrium probabilities

$p_{i,r:j,t}$ the probability of a class r job at the i-th service center transiting to the j-th service center and becoming a class t job

$p_{i,j}$ transistion probability from the i-th CPU to the j-th device in the i-th ICS or to the SPR, where $0 \leq j \leq s_i$

$p_j = p_{i,j}$, $i=1, \ldots, R$ and $j=0, \ldots, s_i$ when network is balanced

$P[n_{i,j} \geq k] = \sum_{n \ni n_{i,j} \geq k} P(n)$ the marginal probability that device (i,j) is serving k or more jobs

$Q_{i,j}$ the mean queue length of device (i,j)

$\rho_{i,j}$ traffic intensity of device (i,j)

R number of ICSs in the network, which is equal to the number of different classes in the network

$S = (s_0, \ldots, s_R)$ the device allocation vector

s_i number of devices in the i-th ICS (i>0) or the SPR (i=0)

$s = s_i$, $i = 1, \ldots, R$ when network is balanced

$T_{i,j}$ average throughput of device (i,j)

$u_{i,j}$ mean processing rate of device (i,j)

$u_j = u_{i,j}$, $i = 1, \ldots, R$ and $j = 0, \ldots, s_i$ when network is balanced

$W_{i,j}$ mean wait time at device (i,j)

$x_{i,j} = \frac{e_{i,j}}{u_{i,j}}$ relative load factor of device (i,j)

APPENDIX D

SAMPLE SPACE FROM ASSIGN ALGORITHM

S= 2, 6, 11,
P(SPR)= 0.100, 0.250, 0.500, 0.750, 0.900,
TOTAL C OMBINATIONS = 15

1. P(SPR) = 0.100
S = 2

	1	2
P(J)=	0.132	0.768
U(J)=	1.000	1.000

2. P(SPR) = 0.250
S = 2

	1	2
P(J)=	0.182	0.568
U(J)=	1.000	0.020

3. P(SPR) = 0.500
S = 2

	1	2
P(J)=	0.024	0.476
U(J)=	1.000	1.000

4. P(SPR) = 0.750
S = 2

	1	2
P(J)=	0.010	0.240
U(J)=	1.000	0.100

5. P(SPR) = 0.900
S = 2

	1	2
P(J)=	0.029	0.071
U(J)=	1.000	5.000

6. P(SPR) = 0.100
S = 6

	1	2	3	4	5	6
P(J)=	0.276	0.374	0.006	0.125	0.026	0.093
U(J)=	1.000	0.200	0.020	0.100	0.020	0.020

7. P(SPR) = 0.250
S = 6

	1	2	3	4	5	6
P(J)=	0.386	0.083	0.077	0.020	0.037	0.147
U(J)=	1.000	0.2001	0.000	0.200	0.010	0.100

8. P(SPR) = 0.500
S = 6

	1	2	3	4	5	6
P(J)=	0.164	0.139	0.009	0.081	0.071	0.036
U(J)=	1.000	2.000	0.020	5.000	0.200	0.100

9. P(SPR) = 0.750
S = 6

	1	2	3	4	5	6
P(J)=	0.054	0.051	0.025	0.011	0.068	0.041
U(J)=	1.000	0.020	10.000	0.100	0.200	0.500

10. P(SPR) = 0.900
S = 6

	1	2	3	4	5	6
P(J)=	0.061	0.003	0.004	0.001	0.013	0.018
U(J)=	1.000	0.500	0.020	0.500	0.010	2.000

11. P(SPR) = 0.100
S = 11

	1	2	3	4	5	6	7	8	9	10	11
P(J)=	0.376	0.259	0.046	0.138	0.006	0.025	0.020	0.001	0.011	0.013	0.005
U(J)=	1.000	0.500	1.000	0.050	5.000	0.020	10.000	5.000	0.200	1.000	0.010

12. P(SPR) = 0.250
S = 11

	1	2	3	4	5	6	7	8	9	10	11
P(J)=	0.551	0.040	0.017	0.089	0.035	0.007	0.003	0.001	0.005	0.001	0.001
U(J)=	1.000	2.000	10.000	10.000	10.000	2.000	0.010	0.200	0.020	0.200	0.020

13. P(SPR) = 0.500
S = 11

	1	2	3	4	5	6	7	8	9	10	11
P(J)=	0.288	0.042	0.016	0.098	0.007	0.004	0.028	0.008	0.001	0.001	0.007
U(J)=	1.000	1.000	5.000	5.000	0.010	0.010	1.000	1.000	0.100	2.000	1.000

14. P(SPR) = 0.750
S = 11

	1	2	3	4	5	6	7	8	9	10	11
P(J)=	0.088	0.113	0.006	0.013	0.012	0.004	0.006	0.005	0.001	0.001	0.001
U(J)=	1.000	0.020	10.000	0.020	0.100	2.000	0.050	10.000	0.200	2.000	0.020

15. P(SPR) = 0.900
S = 11

	1	2	3	4	5	6	7	8	9	10	11
P(J)=	0.003	0.027	0.024	0.006	0.005	0.020	0.006	0.001	0.006	0.001	0.001
U(J)=	1.000	5.000	0.050	1.000	0.010	1.000	10.000	5.000	0.100	5.000	0.500

BIBLIOGRAPHY

[ADIRI 69] Adiri, I. and Avi-Itzhak, B., "A Time-sharing Queue", Manag. Sci., Vol. 15 No. 1, pp 639-657, 1969.

[ADIRI 71A] Adiri, I., "A Dynamic Time-sharing Queue", J.ACM, VOL. 18 No. 4, pp 603-610, Oct. 1971.

[ADIRI 71B] Adiri, I., "A Note on Some Mathematical Models of time-sharing Systems", J.ACM, VOL. 18 No. 4, pp603-610, Oct. 1971.

[AHO 74] Aho, A. V., Hopcroft, J. E. and Ullman, J. D. "The Design and Analysis of Computer Agorithms" Addison-Wesley Pub. Co., Reading, Mass., 1974.

[ALLEN 78] Allen, A. O. "Probability, Statistics, and Queuing Theory" Academic Press, New York, NY, 1978.

[AVI-IT 73] Avi-itzhak, B. and Heyman, D., "Approximate Queueing Models for Multiprogramming Computer Systems", Opns. Res., Vol. 21 No. 6, pp 1212-1230, Dec. 1973.

[BAASE 78] Baase, S. "Computer Algorithms: Introduction to Design and Analysis" Addison-Wesley Pub. Co., Reading, Mass., 1978.

[BAER 73] Baer, J. L., "A Survey of Some Theoretical Aspects of Multiprocessing", Computer Surveys, Vol. 5 No. 1, pp 31-80, Mar. 1973.

[BASKET 72A] Baskett, F. and Gomez, F. P., "Processor Sharing in a Central Server Queueing Model of Multiprogramming With Applications", Proc. 6-th Annual Princeton Conf. on Info. Sci. and Syst., Princeton, N.J., pp 589-603, Mar. 1972.

[BASKET 72B] Basket, F. and Muntz, R., "Queueing Network Models with Different Classes of Customers", Digest of Papers, COMPCON 72, San Francisco, CA., pp 205-209, Sept. 1972.

[BASKET 75] Basket, F.; Chandy, K.; Muntz, R. and Placaios, F., "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers", J. ACM, VOL. 22 No. 5, pp 248-260, Apr. 1975.

[BASKET 76] Baskett, F. and Smith, A. J. "Interference in Multiprocessor Computer Systems with Interleaved Memory", CACM, Vol. 19, No. 6, pp 327-334, June 1976.

[BHANDA 73] Bhandarkar, D. P. "Analytic Models for Memory Interference in Multiprocessor Computer Systems", Ph.D. Thesis, Carnegie-Mellon University, Sept. 1973.

[BHAT 73] Bhat, U. N. and Nance, R. E., "Dynamic Quantum Allocation and Swap-Time Variability in Time-sharing Operating Systems", Southern Methodist University report No. TR-CP-73009, Apr. 1973.

[BHAT 78] Bhat, U. N., Fisher, M. J. and Shalaby, M. "Aproximation Techniques in the Solution of Queueing Problems", Southern Methodist University Rpt., NTIS #ADA053499, Mar. 1978.

[BRANDW 77] Brandwajn, A. " A Queueing Model of Muliprogramming Computer Systems Under Full LOad Conditions", JACM, Vol. 24, No. 2, pp 222-240, Apr. 1977.

[BRECHT 78] Brechtlein, R. "Comparing Disc Technologies", Datamation, pp 139-150, June 1978.

[BROWNE 73] Browne, J. C., et. al. "The Effect on Throughput of Multiprocessing In a Multiprogramming Environment", IEEE Trans. on Comp., Vol. C22 No. 8, pp 728-735, Aug. 1973.

[BROWNE 75] Browne, J. C., et. al. "Hierarchical Techniques for the Development of Realistic Models of Complex Computer Systems", Proc. of the IEEE, Vol. 63, No. 6, pp 966-975, June 1975

[BUCCI 79] Bucci, G. and Streeter, D. "A Methodology for the Design of Distributed Information Systems", CACM, Vol. 22, No. 4, pp 233-245, Apr. 1979.

[BURKE 56] Burke, P. J. "The Output of a Queueing System", Opns. Res., Vol. 4, No. 6, pp 699-704, Dec. 1956.

[BURKE 72] Burke, P. J. "Output Processes and Tandem Queues", Proc. of the Symposium on Computer-Communications Networks and Teletraffic, Polytechnic Institute of Brooklyn, N.Y., pp 419-428, Apr. 1972.

[BURNET 70] Burnett, G. J. and Coffman, E. G., "A Study of Interleaved Memory Systems", Proc. of AFIPS SJCC, Vol. 36, pp 467-474, Spring 1970.

[BUZEN 71] Buzen, J. P. "Queueing Network Models of Multiprogramming", Ph.D. Thesis, Harvard University, 1971.

[BUZEN 73] Buzen, J. P. "Computational Algorithms for Closed Queueing Networks with Exponential Servers", C.ACM, Vol. 16 No. 9, pp 527-531, Sept. 1973.

[BUZEN 74] Buzen, J. P. and Goldberg, P. S. "Guidelines for the use of Infinite Service Queuing Models in the Analysis of Computer System Performance", Proc. of AFIPS NCC 74, Vol 43, pp 371-374, 1974.

[BUZEN 76] Buzen, J. P. "Fundamental Laws of Computer System Performance", Proc. of the International Symposium on Computer Performance Modeling, Measurement and Evaluation, Harvard Univ., Cambridge, Mass., pp 200-210, Mar 1976.

[BUZEN 77] Buzen, J. P. and Potier, D. "Accuracy of Exponential Assumptions in Closed Queueing Models", Proc. of the SIGMETRICS Conf. on Computer Performance: Modeling, Measurement, and Management, Wash., D.c., pp 53-63, Nov. 1977.

[BUZEN 78A] Buzen J. P. and Denning, P. J. "The Operational Analysis of Queuing Network Models", ACM Computing Surveys, Vol. 10, No. 3, pp 225-262, Sept 1978.

[BUZEN 78B] Buzen, J. P. "A Queuing Network Model of MVS", ACM Computing Surveys, Vol. 10, No. 3, pp 319-332, Sept 1978.

[CADY 72] Cady, G. M. "Computation and Communication Trade-off Studies, An Analytical Model of Computer Networks", Proc. WESCON, Vol. 16 Section 7/2, pp 1-12, 1972.

[CARPEN 79] Cartpenter, R. J. and Sokol, J. "Serving Users With a Local Area Network", Proc. of the Local Area Communications Network Symposium, Boston, Mass., pp 75-85, May 1979.

[CDC 75] CDC "Scope 3.4.4 Multi-Mainframe Operations Programming systems Bulletine", Control Data Corp., St. Paul, Minn., CDC # 60493600, 1975.

[CHAMPI 79] Champine, G. A. "Current Trends in Data Base Systems", IEEE Computer, Vol. 12, No. 5, pp 27-41, May 1979.

[CHAMPI 80] Champine, G. A. "Back-End Technology Trends", IEEE Computer, Vol, 13, No. 2, pp 50-58, Feb. 1980.

[CHANDY 72A] Chandy, K. M. "The Analysis and Solutions for General Queueing Networks", Proc. 6-th Annual Princeton Conf. on Info. Sci. and Syst., Princeton, N.J., pp 224-228, Mar. 1972.

[CHANDY 72B] Chandy, K. M. "Local Balance in Queueing Networks With Independent Servers With Applications to Computer Systems", Proc. 10-th Allerton Conf., University of Ill., Oct. 1972.

[CHANDY 75A] Chandy, K. M., Herzog, U., and Woo, L. "Performance Analysis of Queueing Networks", IBM J. Res. Develop., Vol. 19, No. 1, pp 36-42, Jan. 1975.

[CHANDY 75B] Chandy, K. M., Herzog, U., and Woo, L. "Analysis of General Queueing Networks", IBM J. Res. Develop., Vol. 19, No. 1, pp 43-49, Jan. 1975.

[CHANDY 77] Chandy, K. M., Howard, J. H., and Towsley, D. F. "Product Form and Local Balance in Queueing Networks", JACM, Vol. 24, No. 2, pp 250-263, Apr. 1977.

[CHANDY 78] Chandy, K. M. and Sauer, C. H. "Approximate Methods for Analyzing Queuing Network Models of Computer Systems", ACM Computing Surveys, Vol. 10 , No. 3, pp 281-318, Sept 1978.

[CHANG 79] Chang, H. "Major Activity in Magnetic Bubble Technology", Computer Design, Vol. 13, No. 12, pp 117-125, Nov 1979.

[CHANG 72] Chang, J. H. and Gorenstein, S. "A Disk File System Shared by Several Computers in a Teleprocessing Envirnoment", Proc. of the Symposium on Computer-Communications Networks and Teletraffic, Polytechnic Institute of Brooklyn, N.Y., pp 177-185, Apr. 1972.

[CHANG 66] Chang, W. "A Queueing Model for a Simple Case of Time Sharing", IBM Syst. J., Vol. 5, No. 2 pp 115-125, 1966.

[CHLAMT 80] Chlamtac, I. Franta, W. R., Patton, P. C. and Wells, B. "Performance Issues in Back-End Storage Networks", IEEE Computer Vol. 13, No. 2, pp 18-31, Feb 1980.

[CHIU 75] Chiu, W., Dumont, D., and Wood, R. "Performance Analysis of a Multiprogrammed Computer System", IBM J. Res. Develop., Vol. 19, No. 3, pp 263-271, May 1975

[CHOW 75] Chow, W. M. "Central Server Model for Multiprogrammed Computer Systems with Different Classes of Jobs", IBM J. Res. Develop., Vol. 19, No. 3, pp 315-320, May 1975.

[CHOW 77] Chow, W. M. and Woo, L. "Buffer Performance Analysis of Communication Processors During Slowdown at Network Control", IBM J. Res. Develop., Vol. 21, No. 3, pp 264-272, May 1977.

[CITREN 72] Citrenbaum, R. "AN On-line Model for Computation-Communication Network Analysis and Trade-off Studies", Proc. of the Symposium on Computer-Communications Networks and Teletraffic, Polytechnic Institute of Brooklyn, N.Y., pp 613-634, Apr. 1972.

[COFFMA 68A] Coffman, E. G. Jr. "Analysis of Two Time-Sharing Algorithms Designed for Limited Swapping", J.ACM, Vol. 15 No. 3, pp 341-353, July 1968.

[COFFMA 68B] Coffman, E. G. Jr. and Kleinrock, L. "Feedback Queueing Models for Time-Shared Systems", J.ACM, Vol. 15 No. 4, pp 549-576, Oct. 1968.

[COFFMA 69] Coffman, E. G. Jr. and Muntz, R. R. "Models of Resource Allocation Using Pure Time-Sharing Disciplines", Proc. of the 24-th ACM National Conf., pp 217-228, 1969.

[COFFMA 70] Coffman, E. G. Jr., Muntz, R. R., and Trotter, H. "Waiting Time Distributions for Processor Sharing Systems", J.ACM, Vol. 17 No. 1, pp 123-130, Jan. 1970.

[COFFMA 73] Coffman, E. G. Jr. and Denning, P. J. "Operating Systems Theory", Printice-Hall, Inc., N.J., 1973.

[CONWAY 67] Conway, R. W.; Maxwell, W. L. and Miller, L. W. "Theory of Scheduling", Addison-Wesley, Reading, MA., 1967.

[COOPER 69] Cooper, R. B. and Murray, G. "Queues Served in Cyclic Order", Bell Syst. Tech. J., Vol. 48 No. 3, pp 675-689, Mar 1969.

[COTTON 79] Cotton, I. W. "Technologies For Local Area Computer Networks", Proc. of the Local Area Communications Network Symposium, Boston, Mass., pp 25-45, May 1979.

[COURTO 71] Courtois, P. J. "On the Near-Complete-Decomposability of Networks of Queues and of Stachastic Models of Multiprogramming Computer Systems", Ph.D. Thesis, Carnegie Mellon University, Nov. 1971.

[COX 55] Cox, D. R. "A Use of Complex Probabilities in the Theory of Stochastic Processes", Proc. Cambridge Philos. Soc., Vol. 51, pp 313-319, 1955.

[DISNEY 73] Disney, R. L. and Cherry, W. P. "Some Topics In Queueing Network Theory", Proc. of the Sympoium on Mathematical Methods in Queueing Theory, Kalamazoo, Mich., PP 23-44, May 1973.

[DISNEY 77] Disney, R. L. "Research into Queueing Network Theory", NTIS #ADA047148, Sept. 1977.

[DUDICK 71] Dudick, H. L. and Pack, C. D. "Round Robin Scheduling in a Computer Communication System With Finite Swap-Time and Statistically Multiplexed Arrivals", Proc. ACM-IEEE 2nd Symposium on Problems in the Operations of Data Communication Systems, Oct 1971.

[FELLER 68] Feller, W. "An Introduction to Probability Theory and Its Applications", Vol. I, John Wiley ans Sons Inc., Ney York, N.Y., 1968.

[FENICH 69] Fenichel, R. R. and Grossman, A. J. "An Analytic Model of Multiprogramming Computing", Proc. of AFIPS SJCC, Vol. 34, pp 771-721, Spring 1969.

[FERRAR 78] Ferrari, D. "Computer Systems Performance Evaluation", Prentice-Hall Inc., Englewood Cliffs, N.J., 1978.

[FINCH 59] Finch, P. D. "The Output Process of the Queuing System M/G/1", J. Roy. Stat. Soc., Ser B21, No. 2, pp 375-380, 1959.

[FRANK 72] Frank, H.; Kahn, R. E. and Kleinrock, L. K. "Computer-Communication Network Design-Experience With Theory and Practice", Proc. of AFIPS SJCC, Vol 40, pp 255-270, Spring 1972.

[FRATTA 72] Fratta, L. and Gerla, M. "The Synthesis of Computer Networks: Properties of the Optimum Solution", ACM International Computing Symposium, Venice, Italy, 1972.

[FREEMA 80] Freeman, H. A. "Great Editors's Introduction: Back-End Storage Networks", IEEE Computer, Vol. 13, No. 2, pp 7-8, Feb 1980.

[FUCHS 70] Fuchs, E. and Jackson, P. E. "Estimates of Distributions of Random Variables for Certain Computer Communications Traffic Models" C.ACM, Vol. 13 No. 12, pp 752-757, Dec. 1970.

[FUNG 79] Fung, F. T. and Torng, G. C. "In the Analysis of Memory Conflicts and Bus Contentions in a Multiple - Microprocessor System", IEEE Trans. on Comp., Vol. C28, No. 1, pp 28-37, Jan 1979.

[GAVER 67] Gaver, D. P. "Probability Models for Multiprogramming Computer Systems", J.ACM, Vol. 14, No. 3, pp 423-438, July 1967.

[GAVER 73] Gaver, D. P. and Shedler, G. S. "Processor Utilization in Multiprogramming Systems Via Diffusion Approximations", Opns. Res., Vol. 21, No. 2, pp 569-576, Apr. 1973.

[GAVER 76] Gaver, D. P. and Humfeld, G. "Multitype Multiprogramming Models", Acta Informatica, Vol. 7, pp 111-121, 1976.

[GELENB 73] Gelenbe, E. "The Distibution of a Program in Primary and Fast Buffer Storage", C.ACM, Vol. 16 No. 7, pp 431-434, July 1973.

[GERLA 73] Gerla, M. "The Design of Store-And-Forward (S/F) Networks for Computer Communication", Ph.D., UCLA, 1973.

[GIAMMO 76] Giammo, T. "Extensions to Exponential Queuing Network Theory for Use in a Planning Environment", Digest of Papers, COMPCON 76, Fall, Wash., D.C., pp 156-160, Sept. 1976.

[GORDON 67A] Gordon, W. J. and Newell, G. F. "Closed Queueing Systems With Exponential Servers", Opns. Res., Vol. 15 No. 2, pp 254-265, Apr. 1967.

[GORDON 67B] Gordon, W. J. and Newell, G. F. "Cyclic Queuing Systems with Restricted Length Queues", Opns. Res., Vol, 15 No. 2, pp 266-277, Apr. 1967.

[GREENB 73] Greenberg, I. "Distribution-Free Analysis of M/G/1 and G/M/1 Queues", Opns. Res., Vol. 21 No. 2, pp 629-635, Apr. 1973.

[GROSS 75] Gross, D. "Sensitivity of Queuing Models to the Assumption of Exponentiality", Naval Res. Logistics Quarterly, Vol. 22, No. 2, pp 271-287, June 1975.

[HOOGEN 77] Hoogendoorn, C. H. "A General Model for Memory Interference in Multiprocessors", IEEE Trans. on Comp., Vol C26, No. 10, pp 998-1005, Oct 1977.

[HOAGLA 79] Hoagland, A. S. 'Storage Technology: Capabilities and Limitations", IEEE Computer, Vol. 12, No. 5, pp 12-18, May 1979.

[IDC 76] IDC "Distributed Processing", International Data Corp. Rpt #1763, 214 Third Ave., Waltham, Mass., 02154, Dec. 1976.

[JACKSO 57] Jackson, J. R. "Networks of Waiting Lines", Opns. Res., Vol. 5, No. 4, pp 518-521, Aug. 1957.

[JACKSO 63] Jackson, J. R. "Jobshop-Like Queueing Systems", Manag. Sci., Vol. 10 No. 1, pp 131-142, Oct. 1963.

[JAFARI 77] Jafari, H. and Lewis, T. "A New Loop Structure for Distributed Microcomputing Systems", Proc. of the 1st Annual Rocky Mountain Symposium on Microcomputers Systems, Software and Architecture, Ft. Collins, Col., pp 121-141, Aug 1977.

[KAHN 72] Kahn, R. E. "Resource-Sharing Computer Networks", Proc. IEEE, Vol. 60, pp 1397-1407, Nov. 1972.

[KAMOUN 76] Kamoun, F. and Kleinrock, L. "Analysis of Shared Storage in a Computer Network Environment", Proc. of the Ninth Hawaii International Conference on System Sciences, Honolulu, Hawaii, pp 89-92, Jan. 1976.

[KELLER 72] Keller, T. W.; Towsley, D. F.; Chandy, K. M. and Browne, J. C. "A Tool for Network Design: The Automated Analysis of Stochastic Models of Computer Networks", Digest of Papers, COMPCON 72, San Francisco, CA., pp 151-153, Sept. 1972.

[KIMBLE 73] Kimbleton, S. R. "Computer Systems Models", University of Mich. Rpt., NTIS # AD774678, June 1973.

[KLEINR 64] Kleinrock, L. "Communications Nets: Stochastic Message Flow and Delay", McGraw-Hill, N.Y., 1964.

[KLEINR 67] Kleinrock, L. "Time-Shared Systems: A Theoretical Treatment", J.ACM, Vol. 14 No. 2, pp 242-261, Apr. 1967.

[KLEINR 69] Kleinrock, L. "Models for Computer Networks", Proc. of the International Communications Conf., Boulder, CO., Section 21, pp 9-16 Section 21, June 1969.

[KLEINR 70A] Kleinrock, L. "Analytic and Simulation Methods in Computer Network Design", Proc. of AFIPS SJCC, Vol 36, pp 569-579, Spring 1970.

[KLEINR 70B] Kleinrock, L. "A Continuum of Time-Sharing Scheduling Algorithms", Proc. of AFIPS SJCC, Vol 36, pp 453-458, Spring 1970.

[KLEINR 75] Kleinrock, L. "Queueing Systems Volume I: Theory", John Wiley & Sons, Inc., N.Y., 1975.

[KLEINR 76] Kleinrock, L. "Queueing Systems Volume II: Computer Applications", John Wiley & Sons, Inc., N.Y., 1976.

[KOBAYA 74A] Kobayashi, H. "Application of the Diffusion Approximation to Queueing Networks I: Equilibrium Queue Distributions", JACM, Vol. 21, No. 2, pp 316-328, Apr. 1974.

[KOBAYA 74B] Kobayashi, H. "Application of the Diffusion Approximation to Queueing Networks II: Nonequilibrium Distributions and Applications to Computer Modeling", JACM, Vol. 21, No. 2, pp 459-469, Apr. 1974.

[KOBAYA 75] Kobayashi, H. and Reiser, M. "On Generalization of Job Routing Behavior in a Queueing Network Model", IBM Res. Rpt. RC5252 (#23079), Feb. 1975.

[KOBAYA 76] Kobayashi, H., "A Computational Algorithm for Queue Distributions Via Polyas' Theory of Enumeration", IBM Res. Rpt. RC6154 (#26522), Aug. 1976.

[KOBAYA 77] Kobayashi, H. and Konheim, A. "Queuing Models for Computer Communications System Analysis" IEEE Trans. on Comm., Vol COM-25, No. 1, pp 2-28, Jan 1977.

[KOBAYA 78] Kobayashi, H. "Modeling and Analysis: An Introduction to Systems Performance Evaluation Methodology", Addison-Wesley Pub. Co., Reading, Mass., 1978.

[KONHEI 76] Konheim, A. G. and Reiser, M. "A Queueing Model with Finite Waiting Room and Blocking", JACM, Vol. 23, No. 2, pp 328-341, Apr. 1976

[KRISHN 66] Krishnamoorthi, B. and Wood, P. C. "Time-Shared Computer Operations With Both Interarrival and Service Time Exponential", J.ACM, Vol. 13, No. 3, pp 317-338, July 1966.

[LAM 79] Lam, C. Y., "The Intelligent Memory System Architecture - Research Directions", Mass. Inst. of Tech. Rpt., NTIS #ADA073485, Aug. 1979.

[LAM 77A] Lam, S. S., "Queuing Networks with Population Size Constraints", IBM J. Res. Develop., Vol. 21, No. 4, pp 370-378, July 1977.

[LAM 77B] Lam, S. S., "An Extension of Moore's Result for Closed Queuing Networks", IBM J. Res. Develop., Vol. 21, No. 4, pp 384-387, July 1977.

[LAMPSO 80] Lampson, B. W. and Redell, D. D. "Experience with Processess and Monitors in Mesa", CACM, Vol. 23, No. 2, pp 105-117, Feb. 1980.

[LEE 78] Lee, C.M.B. "System Modeling - Where we are now and Where we have to go", Proc. of 17th Annual Tech. Symp.: Tools for Improved Computing in the 80's, NBS, Gaithersbur;g, MD., pp 211-220, June 1978.

[LEUNG 78] Leung, S. K. "Architecture of a Modular Network - The BANCS Network", Proc. COMPON 78: Computer Communications Networks, Wash. D.C., pp 212-220, Setp. 1978.

[LEWIS 71] Lewis, P. A. and Shedler, G. S. "A Cyclic-Queue Model of System Overhead in Multiprogrammed Computer Systems", J.ACM, Vol. 18 No. 2, pp 199-220, Apr. 1971.

[LIPSKY 77] Lipsky, L. and Church, J., "Applications of a Queueing Network Model for a Computer System", ACM Computing Surveys, Vol. 9 No. 3, pp 205-221, Sept. 1977.

[McCRED 73] McCredie, J. W. "Analytic Models as Aids in Multiprocessor Designs" Proc. Seventh Annual Princeton Conf. on Information Sciences and Systems, Princeton, N.J., pp 186-191, March 1973.

[MCKINN 69] McKinney, J. M. "A Survey of Analytical Time-Sharing Models", Computer Surveys, Vol. 2, No. 2, pp 106-116, June 1969.

[MONAHA 79] Monahan, J. H. "Opportunities and Challenges in the Evolution of Local Area Communications Networks", Proc. of the Local Area Communications Network Symposium, Boston, Mass., pp 15-24, May 1979.

[MOORE 72] Moore, F. R. "Computational Model of a Closed Queueing Network with Exponential Servers", IBM J. Res. Develop., pp 567-572, Nov. 1972.

[MORSE 58] Morse, P. M. "Queues, Inventories and Maintenance", John Wiley & Sons, Inc., N.Y., 1958.

[MUNTZ 72A] Muntz, R. R. "Waiting Time Distribution for Round-Robin Queueing Systems", Proc. of the Symposium on Computer-Communications Networks and Teletraffic, Polytechnic Institute of Brooklyn, N.Y., pp 429-440, Apr. 1972.

[MUNTZ 72B] Muntz, R. R. "Poisson Departure Processes and Queueing Networks", IBM T. J. Watson Res. Center, IBM Rpt.# RC4145, Sept. 1972.

[MUNTZ 73] Muntz, R. R. "Poisson Departure Processes and Queuing Networks", Proc. of the 7th Annual Princeton Conf. on Info. Sci. and Syst., Princeton, N.J., pp 435-440, Mar 1973.

[MUNTZ 74] Muntz, R. R. and Wong, J. W., "Efficient Computational Procedures for Closed Queueing Network Models", Proc. of the Seventh Hawaii International Conference on System Sciences, Honolulu, Hawaii, pp 33-36, Jan. 1974.

[MUNTZ 78] Muntz, R. R. "Queuing Networks: A Critique of the State of the Art and Directions for the Future", ACM Computing Surveys, Vol. 10, No. 3, pp 353-360, Sept 1978.

[NAKARM 71] Nakamura, G. "A Feedback Queueing Model for an Interactive Computer System", Proc. of AFIPS FJCC, Vol. 39, pp 57-64, Fall 1971.

[NOETZE 76A] Noetzel, A. S., "Recent Generalizations of Queueing Networks with Product-Form Solutions", Proc. of the 1976 Conference on Information Sciences and Systems, The John Hopkins University, Balt., Md., pp 549-552, Apr. 1976.

[NOETZE 76B] Noetzel, A. S., "Throughput In Locally Balanced Computer System Models", Brookhaven National Laboratory Rpt., Upton, N.Y., NTIS# BNL-22531, Mar. 1977.

[OMAHEN 72] Omahen, K. and Schrage, L. "A Queueing Analysis of a Multiprocessor System With Shared memory", Proc. of the Symposium on Computer-Communications Networks and Teletraffic, Polytechnic Institute of Brooklyn, N.Y., pp 77-88, Apr. 1972.

[OSTERW 72] Osterweil, L. J. "A Stochastic Model of Multiprocessor Access to an Interleaved Memory", Coloardo University Rpt., NTIS# PB-233695, Oct. 1972.

[OUSTER 80] Ousterhout, J. K., Scelza, D. A. and Sindhu, P. S. "Medusa: An Experiment in Distributed Operating System Structure", CACM, Vol. 23, No. 2, pp 92-104, Feb. 1980.

[PACK 73A] Pack, C. D. "The Effects of Multiplexing on a Computer-Communication System", C.ACM, Vol. 16 No. 3, pp 161-168, Mar. 1973.

[PEARCE 78] Pearce, R. C. and Majithia, J. C. "Analysis of a Shared Resource MIMD Computer Organization", IEEE Trans on Comp, Vol. C27, No. 1, pp 64-67, Jan 1978.

[PETERS 77] Peterson, J. L. "Petri Nets", ACM Computer Surveys, Vol. 9, No. 3, pp 223-252, Sept. 1977.

[PRICE 74] Price, T. G. "Balanced Computer Systems", Stanford University Rpt., NTIS# AD/A001071, Apr. 1974.

[RAU 79] Rau, B. R. "Interleaved Memory Bandwidth in a Model of a Multiprocessor Computer System", IEEE Trans on Comp., Vol. C28, No. 9, pp 678-681, Sept 1979.

[REGIS 73] Regis, R. C. "Multiserver Queueing Models of Multiprocessing Systems", IEEE Trans. on Comp., Vol. C22 No. 8, pp 736-744, Aug. 1973.

[REISER 74] Reiser, M and Kobayashi, H. "Accuracy of the Diffusion Approximation for Some Queueing Systems", IBM J. Res. Develop., Vol. 18 No. 2, pp 110-124, Mar 1974.

[REISER 75] Reiser, M. and Kobayashi, H., "Queuing Networks with Multiple Closed Chains: Theory and Computational Algorithms", IBM J. Res. Develop., Vol. 19, pp 283-294, May 1975.

[REISER 76] Reiser, M. and Kobayashi, H., "On the Convolution Algorithm for Separable Queuing Networks", Proc. of the International Symposium on Computer Performance, Modeling, Measuement and Evaluation", pp 109-117, Mar. 1976.

[ROBERT 70] Roberts, L. G. and Wessler, B. D. "Computer Network Development to Achieve Resource Sharing", Proc. AFIPS SJCC, Vol. 36, June 1970.

[ROOME 74A] Roome, W. D. "Modeling and Design of the Communications and Computation Aspects of Computer Networks", Ph.D. Thesis, Cornell Universiy, 1974.

[ROOME 74B] Roome, W. D. and Torng, H. C. "Modeling and Design of Computer Networks With Distributed Computation Facilities", Proc. 1974 Symposium - Computer Networks: Trends & Applications, National Bureau of Standards, Gaithersburg, Md., May 1974.

[ROSE 78] Rose, C. A. "A Measurement Procedure for Queuing Network Models of Computer Systems", ACM Computing Surveys, Vol. 10, No. 3, pp 263-280, Sept 1978.

[SAATY 73] Saaty, T. L. "Elements of Queueing Theory", McGraw-Hill Book Co., Inc., N.Y., 1961.

[SCHERR 66] Scherr, A. L. "An Analysis of Time-Shared Computer Systems", MIT Press, Cambridge, MA., 1966.

[SCHWAR 79] Schwartz, M. "Thoughput and Time Delay Analysis for a Common Queue Configuration in a Multiprocessor Environment", IEEE Trans. on Computers, Vol C-25, No 12, pp 939-941, Dec 79.

[SEGALL 76] Segall, A., "Basic Models for Dynamic Routing in Computer-Communication Networks", Proc. of the Ninth Hawaii International Conference on System Sciences, Honolulu, Hawaii, pp 77-79, Jan. 1976.

[SHEDLE 73] Shedler, G. S. "A Queueing Model of a Multiprogrammed Computer With a Two Level Storage System", CACM, Vol. 16, No. 1, pp 3-10, Jan 1973.

[SHEMER 67] Shemer, J. E. "Some Mathematic Considerations of Time-Sharing Scheduling Algorithms", JACM, Vol. 14, No. 2, pp 262-272, Apr. 1967.

[SHEMER 69] Shemer, J. E. and Heying, D. W. "Performance Modeling and Empirical Measurements in a System Designed for Batch & Time-Sharing Users", Proc. of AFIPS FJCC, Vol. 35, pp 17-26, Fall 1969.

[SHERMA 72] Sherman, S.; Basket, F. and Browne, J. C. "Trace Driven Modeling & Analysis of CPU Scheduling in a Multiprogramming System", C.ACM, Vol. 15 No. 12, pp 1063-1069, Dec 1972.

[SHUM 76] Shum, A. W. "Queuing Models for Computer Systems with General Service Time Distribution" Ph.D. Thesis, Harvard Univ., Dec 1976.

[SHUM 77] Shum, A. W. and Buzen, J. P. "The EPF Technique: A Method for Obtaining Approximate Solutions to Closed Queuing Networks with General Service Times" in Measuring, Modeling, and Evaluating Computer Systems, H. Beilner and E, Gelenbe (Eds.), North Holland Pub. Co., pp 201-220, 1977.

[SMITH 77] Smith, A. J. "Multiprocessor Memory Organization and Memory Interferences", CACM, Vol. 20, No. 10, pp 754-761, Oct 1977.

[SMITH 79] Smith, A. J. "An Analytic and Experimental Study of Multiple Channel Controllers", IEEE Trans. on Comp., Vol C28, No. 1, pp 38-49, Jan 1979.

[SPIRN 76] Spirn, J. R., "Multi-Queue Scheduling of Two Tasks", Proc. of the International Symposium on Computer Performance, Modeling, Measurement and Evaluation, pp 102-108, Mar. 1976.

[SPIRN 79] Sprin, J. R. "Queuing Networks with Random Selection for Service", IEEE Trans. on S. W. Eng., Vol. SE5, No. 3, pp 287-289, May 1979.

[SPRING 78] Spinger, J. F. "The Distributed Data Network, Its Architecture and Operationk", Proc. COMPON 78: Computer Communications Networks, Wash. D.C., pp 221-228, Setp. 1978.

[STOYAN 78] Stoyan, D. "Queuing Networks -- Insensitivity and a Heuristic Approximation", Proc. of the Seventh Conference on Stockastic Processes and Their Applications, Twente University of Tech., The Netherlands, Aug 1977.

[TAKACS 62] Takacs, L. "Introduction to the Theory of Queues", Oxford University Press, N.Y., 1962.

[THEIS 78] Theis, D. J. "An Overview of Memory Technologies", Datamation, pp 113-131, June 1978.

[THORNT 80] Thornton, J. E. "Back-End Network Approaches", IEEE Computer, Vol. 13, No. 2, pp 10-77, Feb 1980.

[TOOMBS 78A] Toombs, D. "CCD and Bubble Memories: System Implications," IEEE Spectrum, pp 36-39, May 1978.

[TOOMBS 78B] Toombs, D. "An update: CCD and Bubble Memories", IEEE Spectrum, pp 22-30, May 1978.

[VOTAW 73] Votaw, D.F. "A Theory of Storage Sizing", Mitre Corp. Rpt., NTIS #AD-765175, July 1973.

[WALLAC 66] Wallace, V. L. and Rosenberg, R. S. "Markovian Models and Numerical Analysis of Computer System Behavior", Proc. AFIPS SJCC, Vol. 28, pp 141-148, 1966.

[WALLAC 72] Wallace, V. L. "Toward an Algebraic Theory of Markovian Networks", Proc. of the Symposium on Computer-Communications Networks and Teletraffic, Polytechnic Institute of Brooklyn, N.Y., pp 397-407, Apr. 1972.

[WALLAC 73] Wallace, V. L. "Algebraic Techniques for Numerical Solution of Queueing Network Theory", Proc. of the Sympoium on Mathematical Methods in Queueing Theory, Kalamazoo, Mich., PP 295-305, May 1973.

[WARNAR 79] Warnar, R. B. J., Calomeris, P. J. and Recicar, S. A. "Computer Peripherial Memory Systems Forecast", National Bureau of Standards, Wash. D.C., NBS #SP500-45, Apr 1979.

[WATSON 80] Watson, R. W. "Network Architecture Design for Back-End Storage Networks", IEEE Computer, Vol. 13, No. 2, pp 32-48, Feb 1980.

[WANG 78] Wang, J. W. "Queuing Network Modeling Computer Networks", ACM Computing Surveys, Vol. 10, No. 3, pp 333-342, Sept. 1978.

[WELCH 79] Welch, T. A. "Analysis of Memory Hierarchies for Sequential Data Access", IEEE Computer, Vol. 12, No. 5, pp 19-26, May 1979.

[WILEY 77] Wiley, J. M. "Just Enough Queueing Theory", Datamation, pp 87-96, Feb. 1977.

[WILKES 79] Wilkes, M. V. and Wheeler, J. D. "The Cambridge Digital Communication Ring", Proc. of the Local Area Communications Network Symposium, Boston, Mass., pp 47-61, May 1979.

[WONG 78] Wong, J. W. "Queueing Network Modeling of Computer Communication Networks", ACM Computing Surveys, Vol. 10, No. 3, pp 333-342, Sept. 1978.

[WYSZEW 74] Wyszewianski, R. J. "Feedback Queues in the Modeling of Computer Systems: A Survey", University of Michigan Rpt., NTIS # AD782360, May 1974.

[ZUSSMA 77] Zussman, R. "Computer Queueing Analysis On a Handheld Calculator", Computer Design, pp 85-94, Nov. 1977.

☆ U. S. GOVERNMENT PRINTING OFFICE : 1980 340-997/298

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBS SP 500-69	2. Performing Organ. Report No.	3. Publication Date November 1980

4. TITLE AND SUBTITLE

Computer Science and Technology:
An Analytic Study of a Shared Device Among Independent Computing Systems

5. AUTHOR(S)

Alan Mink

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

Same as No. 6

10. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 80-600170

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

Global queueing network performance models are developed for the increasingly important class of computer networks comprising a number of independent computing systems sharing a single resource. An extensive bibliography and survey of prior work relating to this topic are included. Analytic expressions of performance measures for this class of systems are derived from the general theory of multi-class queueing networks, and new computational algorithms for evaluating them are presented that are memory-space efficient (linear vs. exponential) compared with known algorithms for the general theory. This exact analytic model, called the Shared Central Server Model, incurs approximately the same exponential time complexity in its evaluation as do all models based on the general theory; because of this, a simple heuristic approximate model of this class of systems is also presented that is computationally efficient in both time and space. Modular expansion of this class of systems is investigated using the approximate model, and a useful relationship is derived between the number of additional independent computing systems and the incremental increase in capability of the shared resource required to maintain the existing level of system performance.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

Approximate queueing models; computer architecture; modular expansion analysis; performance evaluation; performance modeling; queueing models; queueing networks.

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☒ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

176

15. Price

$5.50